Universidade do Minho
Escola de Engenharia

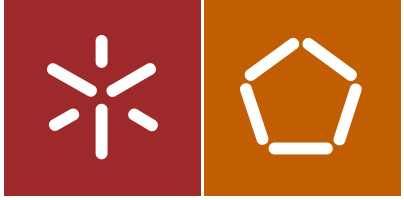
Fábio Joel Sá Lima

Encaminhamento Probabilístico de Dados Nomeados em Redes Tolerantes a Atraso

Encaminhamento Probabilístico de Dados Nomeados em Redes Tolerantes a Atraso
Fábio Joel Sá Lima

UMinho | 2016

janeiro de 2016



Universidade do Minho
Escola de Engenharia

Fábio Joel Sá Lima

Encaminhamento Probabilístico de Dados Nomeados em Redes Tolerantes a Atraso

Dissertação de Mestrado
Ciclo de Estudos Integrados Conducentes ao Grau de
Mestre em Engenharia de Telecomunicações e Informática

Trabalho efetuado sob a orientação de
Professor Doutor Joaquim Melo Henriques de Macedo
Professor Doutor António Luís Duarte Costa

janeiro de 2016

AGRADECIMENTOS

Ao longo do meu percurso académico conheci pessoas fascinantes, que de um modo ou de outro me influenciaram e me tornaram na pessoa que hoje sou. Nesta fase derradeira do meu percurso académico, resta-me deixar-lhes uma palavra de gratidão.

Primeiramente, um agradecimento especial a quem tornou esta dissertação possível. Um agradecimento aos orientadores desta dissertação: Professor Doutor Joaquim Macedo e Professor Doutor António Costa. Um agradecimento pela sua orientação, pela disponibilidade e pelo conhecimento partilhado ao longo deste ultimo ano. Só devido ao seu apoio é que foi possível levar esta dissertação a bom porto.

Gostaria também de deixar uma palavra a todos os meus amigos, tanto os que conheci nestes últimos cinco anos, como os mais antigos. Sem o seu apoio, companheirismo e boa disposição não seria possível concluir esta etapa da minha vida.

Contudo, o maior agradecimento vai para a minha família. Aos meus pais e minha irmã, que me apoiou imenso nestes últimos anos, uma palavra especial. Sem eles, nada disto seria possível e serão alvo da minha gratidão até ao final dos meus dias. Foi graças ao seu esforço, ao facto de prescindirem de muitas coisas, que me permitiram esta oportunidade de ter um percurso académico e ter um futuro melhor.

Por fim, queria ainda deixar uma palavra à minha avó, Felismina Costa.

RESUMO

Atualmente a arquitetura na qual a Internet se baseia assenta na ideia de comunicação fim-a-fim sempre disponível. É esperado que uma ligação entre a origem e o destino da informação esteja sempre disponível. Contudo, existem situações onde tal não é possível, ou não é garantido. Nessas situações é necessário ter uma abordagem diferente.

Foi com esse intuito que nasceram as Redes Tolerantes a Atrasos (*Delay Tolerant Networks*, DTN). Neste tipo de redes, as ligações ponto a ponto são esporádicas, podendo acontecer em determinado momento, e no momento seguinte deixarem de estar disponíveis. Por exemplo, numa rede em que os nós são móveis, quando dois nós se cruzam estabelecem ligação. No momento seguinte, afastam-se e a ligação perde-se. As DTNs têm a capacidade de operar neste tipo de cenários, por exemplo.

Para além desta assunção de que existe sempre uma ligação fim-a-fim disponível, a Internet, atualmente baseia-se também no conceito de IPs, ou seja, endereço de origem e endereço de destino. A informação em si não é tida em conta. Ultimamente tem sido tentada uma abordagem diferente. Uma abordagem na qual é tido em conta o conteúdo das mensagens. Um exemplo deste tipo de redes são as Redes de Dados Nomeados (*Named Data Network*, NDN). É aplicado o paradigma de produtor/consumidor, no qual existem dois tipos distintos de nós. O primeiro cria a informação e o segundo consome-a. Um nó que produza determinado tipo de informação coloca-a na rede, os nós que tiverem interessados na mesma, devem então subscrevê-la.

O principal objetivo desta dissertação é acoplar estes dois tipos de redes. Mais precisamente tentar colocar uma abordagem semelhante às NDNs em cenários onde se utilizam DTNs. Nomeadamente, é sugerida uma abordagem no que diz respeito à distribuição de dados nomeados em cenários onde as DTNs operam.

Pretende-se melhorar um protocolo de encaminhamento probabilístico que encaminha mensagens tendo em conta encontros prévios. Transportou-se para as redes de dados nomeados o paradigma das redes oportunistas de que se dois nós se encontram a probabilidade de se voltarem a encontrar aumenta. Havia-se aplicado esse conceito no contacto com conteúdos, agora esse conceito é também aplicado no contacto de interesses. Ou seja, os conteúdos são agora encaminhados consoante a probabilidade do nó recetor encontrar interessados no mesmo.

ABSTRACT

Nowadays the Internet is based on the idea that end-to-end link is always available. It is expected that a link between the source and destination is always available. But there are some scenarios where that is not possible our at least is not granted. In these situations a different approach is necessary.

For that reason the Delay Tolerant Network (DTN) was born. In this kind of networks, links between nodes are exporadic, at some moment they are available, but after that they can no longer exist. For instance, in a mobile network, when nodes get physically close to each other they connect. In the next moment, one of the nodes move and the connection is lost. DTNs have the capacity to work in this kind of scenarios, for example.

Besides this assumption, that an end-to-end link is always available, the Internet is based in the IP concept, that is, source and destination addresses. The information itself is not taken into account. Lately, a new approach has been tested. An approach where the content inside the messages is taken into account. One example of this kind of network is the Named Data Networks (NDN). A publisher/subscriber paradigm is applied. There are two different types of nodes. The first type creates information and the second one consumes it. A node that creates information, makes it available to the network, the ones that are interested in that information, should subscribe it.

The main goal of this thesis is to combine this two types of network. More precisely, try to apply a NDN approach to a DTN scenario. To do this, a new named content distribution in DTN scenarios approach is suggested.

Is intended to improve a probabilistic routing protocol that forwards messages based on previous encounters. The idea that if one node meets another node they are likely to meet again in the future, was transported from the oportunistic concept to the named data concept. That paradigm was first applied in contacts to contents, now it is also applied to interest contacts. That means, contents are now forwarded based on the other node's probability to find interest in that content.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Pseudo-Códigos

Acrónimos

Nome	Significado
ADU	Aplication Data Unit
BOND	Broadcast-Only Named Data
CCND	Content Centric Network Daemon
CEDO	Content-Centric Dissemination Algorithm for Delay-Tolerant Networks
CLA	Convergence Layer Adapters
COBRA	COntent-driven Bloom filter based Routing Algorithm
DONA	Data-Oriented Network Architecture
DTN	Delay/Disruption Tolerant Networks – Redes Tolerantes a Atrasos e Interrupções
DTNRG	Delay Tolerant Networking Researsh Group
EID	EndPoint Identifiers
FIB	Forwarding Information Base
FIFO	First In First Out
ICDTN	Information-Centric Delay Tolerant Network
ICN	Information Centric Networking
ICONE	Information Centric ONE
LFBL	Listen First, Broadcast Later
LFU	Least Frequently Used
LRU	Least Recently Used
LSA	Link State Advertisments
LSDB	Link-State Data Base
MANET	Mobile Ad-hoc Network – Redes móveis sem infraestrutura
MRG	Minimum Reception Group
NAIF	Neighborhood-aware Interest Forwarding
NDN	Named Data Network – Rede de dados nomeados
NLSR	Named-data Link State Routing Protocol
OLSA	Opaque Link Sate Advertisments
ONE	Opportunistic Network Environment
OSPFN	Open Shortest Path First for Names
PDU	Protocol Data Units
PIFP	Probabilistic Interest Forwarding Protocol
PIT	Pending Interest Table – Tabela de Pedidos Pendentes
PROPHET	Probabilistic Routing Protocol for Intermittently Connected Networks
PURSUIT	Publish-Subscribe Internet Technologies
RTT	Round-trip time
STCR	Social-Tie based Content Retrieval
TCP/IP	Transmission Control Protocol/Internet Protocol

TTL	Time-to-Live
URI	Uniform Resource Identifier
XIA	eXpressive Internet Architecture

Capítulo 1 - Introdução

Hoje em dia a Internet é o principal veículo de propagação de informação. Milhares de informações são produzidas diariamente e colocadas a circular na Internet. Livre, rápida, robusta e de fácil acesso, este é o meio de comunicação preferido quer por quem gera informação, quer por quem a procura. Contudo, em determinadas situações, a sua robustez não é suficiente e são necessárias soluções mais eficazes.

Por vezes, existem cenários em que a conectividade fim-a-fim não é garantida ou não é estável. Situações com grande atraso na entrega da informação, com elevado número de erros, com baixa taxa de entregas ou em que o racionamento de energia dos dispositivos é fulcral, requerem outra arquitetura e protocolos que não os utilizados na Internet [1]. Exemplo de redes com estes problemas, ou pelo menos com alguns deles são as redes sem fios, comunicação de dispositivos móveis, comunicação em áreas remotas (por exemplo minas, desertos, montanhas, zonas rurais, etc…). Podem ser também incluídas comunicações em cenários de guerra ou catástrofe, uma vez que as infra-estruturas de apoio foram destruídas ou danificadas.

Para garantir a comunicação neste tipo de cenários, foi criado o conceito de Rede Tolerante a Atrasos (*Delay Tolerant Network, DTN*). Na arquitetura convencional utilizada na implementação da Internet, a norma é a existência de uma conexão fim-a-fim. A exceção é quando tal não acontece devido a um qualquer motivo. Já nas DTNs a norma é que a comunicação fim-a-fim não exista e a exceção dá-se quando tal acontece. Posto isto, as DTNs necessitam de uma outra abordagem no tratamento da informação, uma vez que a conexão fim-a-fim é algo que não existe ou pelo menos é algo que não é garantido [2]. Para lidar com este tipo de dinamismo, no qual surgem e desaparecem conexões com bastante frequência, as DTNs baseiam-se no conceito *store-carry-and-forward* [3]. Isto é, um determinado nó recebe dados e pode não os conseguir enviar no mesmo momento para o destino. Os dados são guardados na sua *cache* e o nó move-se, até que por fim encontra a possibilidade de o entregar ou de o reproduzir. Ou seja, caso o novo nó não seja o destino final, este recebe uma cópia da mensagem, aumentado a probabilidade de entrega [3].

Por exemplo, foi colocada uma sonda no oceano Atlântico para monitorizar a temperatura da água. Uma vez que naquele local não existe cobertura de rede móvel e que a poupança de

energia é fulcral, os dados não podem ser enviados no momento em que são recolhidos. Sendo assim, a sonda aguarda até que passe por si um navio. Os dados são recebidos pelo navio e este guarda-os e transporta-os até à costa onde, aí sim, podem ser enviados até ao seu destino final e convenientemente processados. Contudo, este tipo de comunicação tem bastantes limitações. Por exemplo o tamanho da mensagem a enviar, a largura de banda da ligação, o tempo da ligação, a frequência com que determinada ligação se estabelece, o tamanho da *cache* de cada um dos nós, etc…

Uma outra abordagem tem sido tentada ultimamente nas DTNs. Esta nova abordagem baseia-se na informação e não nos nós final e inicial e nas suas posições. Isto é, esta nova abordagem tem em conta a informação e dependendo do seu conteúdo, é encaminhada de determinada forma [4]. Na abordagem corrente, o envio dos pacotes é feito tendo em conta o endereço inicial e final, sem que a informação seja tida em conta. Nesta nova abordagem, é aplicado o paradigma de publica/subscreve. Este paradigma descreve como que uma rede centrada na informação. Isto é, o seu endereçamento é feito tendo em conta o conteúdo dos pacotes. Uma cópia de cada um deles vai sendo armazenada localmente à medida que se encaminham para o seu destino, se assim for possível. Isto permite que a informação seja reutilizada e encaminhada de uma forma mais eficiente tendo em conta as necessidades da rede [5]. Um qualquer nó que produz determinado tipo de conteúdo coloca-o na rede os nós que estiverem interessados nessa informação subscrevem-na. Desta forma, vários nós podem receber a mesma informação sem que esta tenha que ser reenviada várias vezes por quem a produz.

Um exemplo deste tipo de redes são as NDNs (*Named Data Network*). Neste tipo de redes o enfase é dado à informação, ao conteúdo de cada um dos pacotes enviados e não à sua origem e destino como acontece hoje em dia. Esta abordagem é considerada uma das mais promissoras para a Internet do futuro [6]. A junção destes dois tipos de rede – DTNs e NDNs – é o objetivo principal deste trabalho. Apesar de serem tipos de redes distintas, com diferenças consideráveis, são também dois tipos de redes com alguns pontos em comum, nomeadamente o facto de utilizarem a *cache* dos dispositivos para armazenamento de informação, a longevidade dos dados, o encaminhamento flexível, entre outros [4].

Tendo isto em conta, o facto desta nova arquitetura se focar na informação e não nos nós e ao ser aplicada nas DTNs pode revelar enormes melhorias na forma como estas operam. Nas DTNs devido a falta de espaço na *cache*, ou por um outro qualquer motivo, os nós descartam informação de forma egoísta. Ou seja, não é tida em conta a informação que está a ser descartada, sendo que essa informação poderia ser importante para um nó vizinho. Se se aplicar

esta nova abordagem das NDNs nas DTNs melhorias a nível de taxa de entrega, de velocidade de entrega e até mesmo de taxa de erros podem ser alcançadas.

Todo o processo de testes que serão feitos nesta dissertação utilizará o simulador "ICONE". Esta é uma ferramenta, desenvolvida em Java, que tem por base o simulador de DTNs "The One" [7]. Este simulador é também ele desenvolvido em Java e permite simular e estudar o comportamento das DTN's. Foram aplicadas algumas alterações e implementadas algumas extensões para que o "The One" tivesse a capacidade de lidar com o paradigma proposto pelas NDNs, resultando assim no "ICONE".

Como dito anteriormente o objetivo deste trabalho é agregar estes dois tipos de rede e testar até que ponto existem melhorias no funcionamento das DTNs. Para tal, será dada continuidade ao trabalho desenvolvido anteriormente noutras dissertações [8], [9].

1.1. Metodologia

A primeira fase desta dissertação consiste no estudo do estado da arte. Isto é, será feita uma revisão da literatura acerca das DTNs e das NDNs por forma a perceber o funcionamento destes dois tipos de redes. É importante perceber até que ponto estas duas redes se diferenciam. Ter uma noção acerca de quais os mecanismos e protocolos necessários implementar nas DTNs para que seja então possível introduzir dados nomeados, bem como o conceito de publica/subscreve. Um outro objetivo desta primeira fase é perceber em que estado de evolução se encontram estes dois tipos de rede. Rever os últimos estudos e trabalhos acerca destes mesmos temas é algo importante para perceber o estado da evolução.

Após esta primeira fase, serão feitos testes e várias experiências com o ambiente de simulação ICONE, tendo como objetivo a familiarização com esta ferramenta. O ICONE é um ambiente de simulação, desenvolvido em Java, que tem por base o simulador de DTNs "The ONE", também ele implementado na linguagem de programação Java [7]. O ICONE permite que sejam agregadas características das NDNs ao simulador "The ONE". Ou seja, permite a simulação de DTNs conjuntamente com as NDNs. O objetivo principal desta dissertação é desenvolver e completar o ICONE.

Para tal, serão implementadas melhorias no protocolo PIFP. Este protocolo foi anteriormente desenvolvido, contudo ainda não se encontra totalmente funcional. Apenas está completa a comunicação no sentido interessado-produtor. Nesta fase, deverá ser desenvolvida a comunicação no sentido inverso, utilizando o mesmo conceito. Será utilizado o protocolo

anteriormente desenvolvido e apenas será adicionada esta funcionalidade, esperando que desta forma a eficiência deste protocolo aumente.

Em seguida, pretende desenvolver-se uma forma de fragmentar as mensagens a circular na rede. Deverá ser desenvolvido um algoritmo que permita que as mesmas sejam como que cortadas para que o seu tamanho seja mais reduzido. Desta forma, a sua transmissão tornar-se-á mais célere. Nesta fase deverão ser tidas em conta algumas características da rede, nomeadamente o tempo médio de ligação entre os nós, a largura de banda das ligações, etc…

Uma outra componente desta dissertação são os algoritmos de *caching,* nomeadamente deverá proceder-se à sua melhoria e posterior implementação, uma vez que estes foram anteriormente abordados, contudo ainda não se encontram muito completos. Esta é uma área a ser explorada nesta dissertação, uma vez que a eficiência destes protocolos pode ditar uma melhoria no funcionamento da plataforma ICONE. Serão corrigidos possíveis erros existentes nos algoritmos até agora desenvolvidos. Não obstante, poderão ser desenvolvidos novos algoritmos de *caching* se assim se justificar.

Por fim, serão feitos testes para verificar até que ponto estas melhorias implementadas no simulador ICONE se verificam. Serão recolhidos dados acerca das várias simulações, utilizando os diversos modelos de mobilidade implementados nas DTNs.

1.2. Objetivos

Depois do enquadramento acima e tendo em conta o trabalho já realizado anteriormente, é então possível então apontar como objetivo principal o desenvolvimento do encaminhamento no sentido produtor – interessado (Melhoria do protocolo PIFP).

Pretende-se concluir trabalhos já realizados. Ou seja, em trabalhos anteriores foi desenvolvido o protocolo PFIP (*Probabilistic Interest Forwarding Protocol*) [8]. Este protocolo é baseado no protocolo de encaminhamento em DTNs PROPHET [10], o qual utiliza a "história" de um nó e calcula a probabilidade de este se voltar a encontrar com determinado nó. Contudo, o protocolo PIFP não calcula a probabilidade de dois nós se encontrarem mas sim a probabilidade de um nó produtor de determinada informação se cruzar com um nó interessado nessa informação. Posto isto, até ao momento apenas foi desenvolvido encaminhamento no sentido interessado-produtor, nesta dissertação, o objetivo é implementar o mesmo conceito no sentido inverso, ou seja, no sentido produtor-interessado. (Se determinado nó se cruza várias vezes com outros nós interessados no

conteúdo “A”, então esse nó é um bom nó para encaminhar informação com o conteúdo “A”).

1.3. Resultados Obtidos

O principal contributo desta dissertação é a melhoria do protocolo de encaminhamento PIFP. Este protocolo, baseado no protocolo de encaminhamento para Redes Tolerantes a Atrasos, PRoPHET, explora apenas a frequência com que um determinado nó contacta com diferentes conteúdos. Agora, para além de explorar esses contactos com os conteúdos, é capaz também de explorar a frequência com que um nó contacta com interesses num determinado conteúdo.

Um outro contributo é o estudo feito sobre até que ponto as Redes de Dados Nomeados podem ser aliadas às Redes Tolerantes a Atrasos. É importante perceber até que ponto uma arquitetura baseada nos nomes dos conteúdos pode ser aplicada a cenários onde apenas uma arquitetura como as DTNs consegue operar.

1.4. Estrutura do Documento

Por forma a facilitar a consulta do documento, nesta secção é fornecida uma visão geral da estrutura do mesmo.

Primeiramente é feita uma introdução ao tema, na qual é apresentado um enquadramento, quais os objetivos que se pretendem alcançar e qual a metodologia de trabalho utilizada. Nos três capítulos seguintes é feito um levantamento do estado da arte. Ou seja, são expostos os resultados da pesquisa efetuada.

Assim sendo, no capítulo dois são discutidas as características principais das Redes Tolerantes a Atrasos. São introduzidos conceitos e componentes subjacentes a esta arquitetura. No capitulo três são apresentadas as características e introduzidos os componentes das Redes de Dados Nomeados, bem como todos os conceitos associados a esta arquitetura. Por fim, no capítulo quatro é feito um levantamento das diferenças e dos pontos em comum destas duas arquiteturas. É ainda apresentado um conjunto de trabalhos já realizados tendo em vista a aproximação das mesmas.

No capítulo cinco é feito um resumo conceptual do trabalho anteriormente desenvolvido na implementação da plataforma ICONE. Para além disso, é também apresentada uma explicação conceptual das melhorias introduzidas no desenvolvimento desta dissertação.

No sexto capítulo é descrita a implementação feita em trabalhos anteriores, sendo feita também uma descrição pormenorizada da implementação desenvolvida ao longo desta dissertação para que as melhorias sugeridas pudessem ser atingidas.

No capítulo sete são apresentados os testes realizados, bem como os resultados obtidos. Neste capítulo, é também feita uma análise crítica dos resultados e é feita uma comparação com as versões anteriores.

Por fim, no oitavo e último capítulo, são apresentadas as conclusões e são sugeridos aspetos a melhorar no futuro.

Capítulo 2 – Rede Tolerante a Atrasos

2.1. Introdução

Nos dias de hoje, a Internet serve como um dos principais meios de comunicação. Milhares de informações são partilhadas utilizando esta tecnologia, constituindo um dos principais, senão mesmo o principal, meio de partilha de informação. Utilizada por grande parte da população mundial, atualmente a Internet apresenta uma grande robustez, existindo muito raramente falhas sentidas pelo utilizador.

Posto isto, existe hoje uma grande preocupação em fazer com que a informação chegue a todo o lado. Como a Internet é um dos principais meios de transporte de informação, é fundamental que tenha um bom funcionamento em todos os cenários de utilização. Contudo, tal não acontece. Atualmente, a arquitetura da Internet é baseada nos protocolos TCP/IP e apesar de não ser explicito, é baseada também num conjunto de pressupostos [1], [3]:

- Existe uma ligação ponto-a-ponto entre a origem e o destino que dura, no mínimo, o mesmo tempo que uma sessão de comunicação;
- As perdas de informação fim-a-fim são relativamente pequenas;
- O tempo máximo de ida e volta (*Round-trip time, RTT*) não é excessivo;
- Todos os encaminhadores e todos os nós suportam protocolos TCP/IP;
- As aplicações não precisam de se preocupar com a performance da ligação;
- Mecanismos de segurança apenas no nó de destino são suficientes para preencher todos os requisitos de segurança;
- Escolher um único caminho entre a origem e o destino é suficiente para atingir uma boa performance de comunicação;
- Infraestruturas acessíveis (servers, encaminhadores, etc…).

Existem certos cenários onde uma ou mais destas condições não se verificam. Estas situações estão a ser cada vez mais tidas em conta e começam a ser uma regra e não uma exceção, como acontecia anteriormente. Nestes cenários, onde não se verificam todas as condições acima referidas, a Internet, baseada numa arquitetura TCP/IP, não funciona

corretamente, ou não consegue funcionar de todo. Existem situações em que os problemas advêm das ligações: atrasos muito longos, ligações de baixa velocidade (comunicações subaquáticas, por exemplo), elevada taxa de erros (redes sem fios, redes subaquáticas, redes de satélites, etc…). No fundo, casos onde a comunicação pode não ser possível/eficiente ou confiável. Situações em que os problemas advêm do alcance e da densidade dos nós: nós com mobilidade previsível (autocarros, comboios, etc…), nós com mobilidade semi-previsível (redes de sensores dispersos, por exemplo) e nós com mobilidade imprevisível, como por exemplo animais selvagens.

Um exemplo é um ambiente de mobilidade terrestre, ou seja, um ambiente em que os nós se encontram em movimento e as ligações nem sempre estão disponíveis e quando estão, não se sabe por quanto tempo. Um fator determinante neste tipo de situações são as interferências no sinal de cada um dos nós. Estas interferências diminuem a força do sinal, levando a que a distância máxima para o estabelecimento de contacto seja mais curta [1]. Outro exemplo são as redes exóticas, como por exemplo redes de comunicação de satélites ou ligações acústicas por ar ou água. Este tipo de redes podem estar sujeitos a grandes latências ou interrupções que podem ser ou não previstas. Isto devido, não só à grande distância entre os nós, como também às condições do ambiente onde se encontram (clima, por exemplo) [1]. Podem também existir redes em que a energia dos nós é limitada e como tal é um fator importante a ter em conta. Neste tipo de redes, normalmente, os nós encontram-se desligados a maior parte do tempo, interrompendo assim as ligações que haviam estabelecido, por forma a poupar energia. As redes de sensores sem fios são um exemplo deste tipo de redes. Um outro exemplo são as Redes Moveis Ad-hoc (MANETs), nas quais não existem infraestruturas físicas e são os terminais móveis que atuam como encaminhadores e encaminham os pacotes de dados. Neste tipo de redes é necessário uma densa cobertura de nós, uma vez que é assumido que existe sempre um caminho disponível entre qualquer emissor e qualquer recetor. As MANETs falham quando tal não acontece, ou seja, quando não existe uma ligação fim-a-fim [11].

As redes que funcionam nos cenários acima descritos têm características únicas e é bastante complicado fazer com que comuniquem entre si. Para permitir que a informação flua entre estes tipos de redes diferentes das redes convencionais, surgiu um novo tipo de rede denominado de Rede Tolerante a Atrasos (*Delay Tolerant Network*, DTN). Este novo tipo de rede suporta atrasos longos e de duração variada. Para além disso, é capaz de lidar com as frequentes desconexões que podem acontecer em situações como as descritas anteriormente. Ou seja, as DTNs são capazes de operar onde os protocolos baseados em TCP/IP, utilizados para a implementação da arquitetura da Internet, falham.

2.2. Arquitetura das DTNs

A arquitetura das DTNs é uma arquitetura para uma rede de sobreposição, ou seja, é uma arquitetura que funciona acima da pilha protocolar das diversas redes [1]. Desta forma, é possível a ligação entre redes com pilhas protocolares diferentes. Por exemplo, no caso da Internet, a DTN pode funcionar sobre TCP/IP.

Como a DTN tem como objetivo funcionar sobre arquiteturas protocolares diferentes, garantindo a comunicação entre nós DTN suportados por diferentes redes é necessário a existência de *gateways* DTN [1]. Por exemplo um *gateway* DTN permite a comunicação entre o nó A na Internet (TCP/IP) e o nó B, que é um sensor numa rede subaquática.

Para além de manipular os pacotes DTN e encaminhá-los para o nó DTN de destino, que são as funções dum encaminhador DTN, os *gateways* desempacotam esses pacotes dos invólucros originais (na sub-rede original) e colocam-nos no invólucro da sub-rede de saída.

Para comunicação entre nós DTN suportados na mesma rede, basta a existência de encaminhadores DTN.

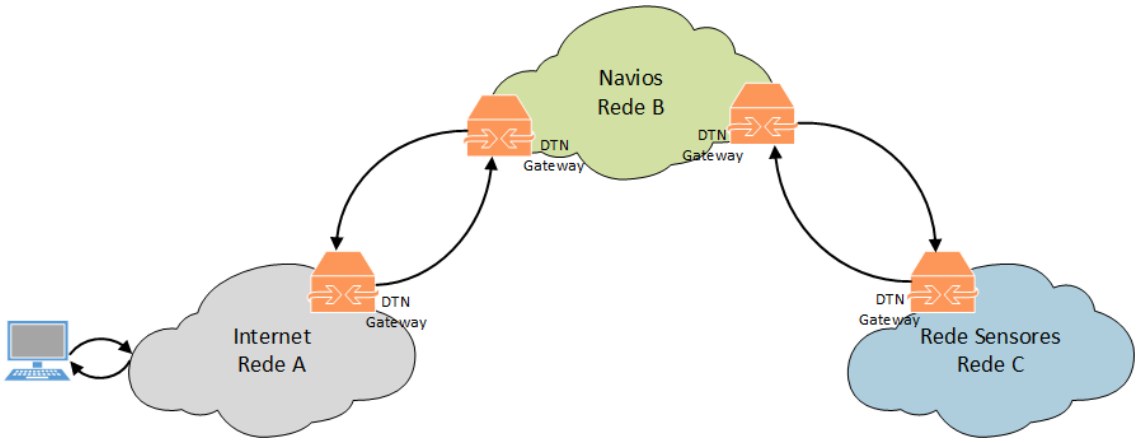


Figura 1 - DTN Gateways interligando várias Redes (adaptado de [1])

Na Figura 1 podemos ver um exemplo em que uma rede de sensores sem fios, colocada no oceano para medir a sua temperatura, tenta enviar os dados para a estação central, a fim de serem processados e analisados. Para efetuar tal comunicação é necessário que a rede de sensores sem fios, que utiliza a sua própria pilha protocolar, estabeleça uma ligação com algum dos navios que passam junto a si. Por sua vez, os navios utilizam uma outra pilha protocolar para comunicarem entre si. Finalmente, os navios devem encaminhar a informação para a estação

central através de uma terceira rede, à qual está ligado o destino final dos dados. A comunicação entre estes três tipos de rede só é possível devido aos DTN *gateways* colocados no ponto de intercomunicação de cada uma das redes.

Para além de ser uma rede de sobreposição e de conseguir operar em cenários mais adversos, existem ainda outras diferenças entre as DTNs e as redes TCP/IP convencionais. Uma delas é a introdução do conceito de *store-carry-and-forward*[1]. Este conceito indica que se um nó não poder encaminhar imediatamente os dados, deve então guardá-los e mover-se como anteriormente. Quando estabelecer um contacto com um outro nó, deve então encaminhar os dados que haviam sido guardados. Ou seja, é esperado que se as ligações não se encontrarem disponíveis ou não forem de confiança, os nós escolham guardar os dados durante algum tempo. Nas redes convencionais, se um nó não for capaz de encaminhar os dados, quer porque a ligação não existe, quer por qualquer outro motivo, estes são descartados. AS redes TCP/IP baseiam-se na assunção de que não é necessário guardar os dados por um período maior do que o tempo gasto para colocar os dados na fila de encaminhamento ou que o atraso de transmissão, ou seja, um período de tempo na ordem dos milissegundos.

Recorrendo novamente ao exemplo da rede de sensores sem fios que tenta comunicar com a central apresentado anteriormente, pode verificar-se como é que este conceito funciona num ambiente real.

[1] Em português: “guarda-transporta-e-encaminha”

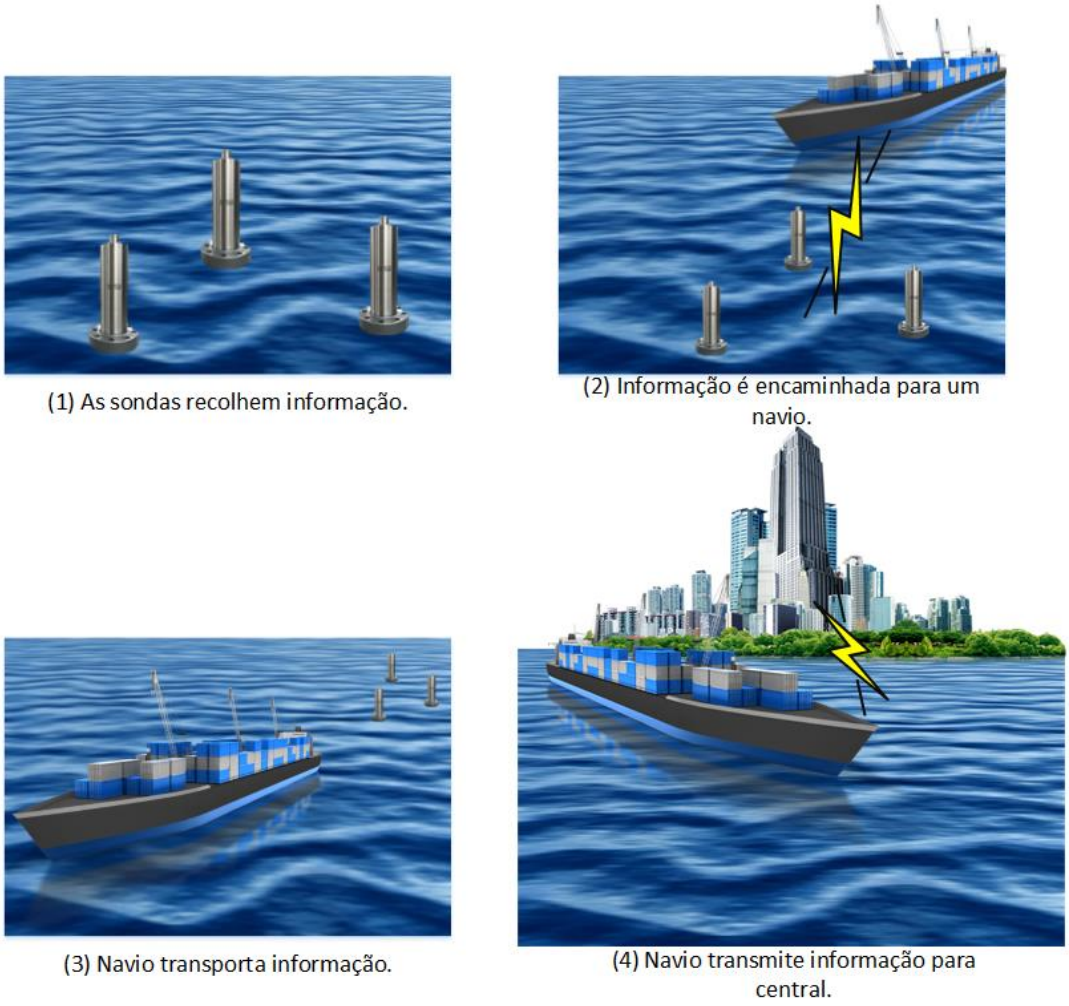


Figura 2 - Exemplo de um cenário DTN

Uma vez que as sondas se encontram no meio do oceano, o fator energia é um fator de extrema importância. É necessário que as sondas poupem o máximo de energia que conseguirem, de forma a estarem ativas o maior tempo possível, sem que tenham de ser substituídas. Para tal, as sondas encontram-se desligadas por grandes períodos de tempo. Apenas são ativadas para recolher informação e enviá-la assim que se conectarem com um qualquer outro nó. Posto isto, as sondas recolhem informação e têm de a enviar para a central terrestre para que possa ser analisada. Como não existe cobertura de rede móvel e a energia deve ser poupada ao máximo, enviar a informação por satélite iria ser muito dispendioso. Sendo assim, as sondas não têm qualquer forma de encaminhar a informação e esta é guardada. As sondas esperam que passe algum navio para procederem ao respetivo encaminhamento. Assim que um navio passa por perto é estabelecida uma ligação e as informações recolhidas pelas sondas são enviadas para o navio. Tal como as sondas, o navio não é capaz de enviar

imediatamente as informações até ao seu destino. Deve então guardar as informações e seguir a sua rota. Assim que chega a terra, o navio é capaz de despachar as informações que haviam sido guardadas anteriormente. Deste modo, as informações são finalmente entregues no destino final onde podem ser convenientemente analisadas. Contudo, este conceito não é aplicado apenas em situações de isolamento. Isto é, mesmo em situações em que os nós se encontram próximos uns dos outros, por vezes não é possível encaminhar as mensagens para o próximo nó.

Tanto numa situação como noutra, é bastante importante que a gestão do *buffer* seja eficiente, uma vez que deve perder-se o menor número de mensagens possível. O controlo de congestão torna-se bastante complicado nestas situações. Estes problemas serão abordados mais adiante nesta dissertação.

Para que este conceito de *store-carry-and-forward* possa ser aplicado, devem ser consideradas as seguintes assunções [3]: espaço para guardar as mensagens está sempre disponível e bem distribuído ao longo da rede; Armazenamento é suficientemente persistente e robusto para guardar as mensagens até que estas sejam enviadas; Esta aproximação de *store-carry-and-forward* é a melhor escolha que tentar efetuar persistentemente uma conexão ou qualquer outra alternativa.

2.3. Protocolo Bundle

Tal como referido anteriormente, as DTNs têm a capacidade de operar em situações extremas. Desde ambientes bastante remotos onde a densidade dos nós é bastante baixa, levando assim a que exista uma grande distância entre os mesmos, a ambientes em que as ligações são quase inexistentes ou bastante intermitentes, passando por situações onde existe uma elevada taxa de erros. Para além de tudo isto, as DTNs são capazes de assegurar a comunicação entre redes que utilizam a sua própria pilha protocolar intra-rede.

Por forma a ter toda esta tolerância e versatilidade, as DTNs utilização o conceito de *Bundle Protocol*. Este é um protocolo orientado à mensagem que, na pilha protocolar, se situa logo abaixo da camada Aplicação [12]. Esta camada, que é implementada acima da camada de transporte da pilha protocolar utilizada numa qualquer rede, permite então que sejam utilizados os protocolos que melhor se adaptam a cada uma dessas mesmas redes. O *Bundle Protocol* funciona então como que uma ponte entre as diferentes pilhas protocolares [13]. Desta feita, é possível tirar o máximo partido de cada uma das redes, uma vez que, utilizam o melhor

protocolo possível intra-rede e estão interligadas entre si graças à nova camada implementada pelas DTNs.

Uma vez que neste tipo de redes os atrasos podem ser arbitrariamente longos, poderá existir uma elevada taxa de erros ou as ligações poderão ter velocidades reduzidas, as comunicações entre os nós origem e destino não poderão conter qualquer tipo de negociações. Ou seja, não poderão ser baseadas em qualquer tipo de linha temporal. Não é prudente que se efetuem comunicações em que é necessário esperar pela confirmação de uma mensagem para se enviar a próxima. Posto isto, é necessário que o "pacote" enviado contenha não só a informação da aplicação – a informação do utilizador – mas também contenha toda a metadata necessária para satisfazer o pedido [13]. Estes "pacotes", que juntam informação útil com metadados necessários, são denominados de "*bundles*".

Resumidamente, nas DTNs as aplicações têm a possibilidade de enviar mensagens de tamanho arbitrário. Essas mensagens são chamadas de "*Aplication Data Units*[2]" (ADUs). Na camada de agregação (do inglês: *Bundle Layer*) essa informação é encapsulada em *Protocol Data Units*[3] (PDU) denominadas "*bundles*". Por fim, os *bundles* são encaminhados pelos nós DTN. Um *bundle* tem um formato definido e podem conter dois ou mais "blocos" de dados. Cada bloco pode conter tanto informação útil, como também informação necessária para fazer chegar o *bundle* ao destino. Durante a transmissão, um *bundle* pode ser fragmentado, ou seja, dividido em dois ou mais *bundles*. Tal pode acontecer devido a vários fatores que serão discutidos mais adiante. Cada um desses fragmentos constitui um novo *bundle*, sendo que em qualquer outro nó, dois ou mais *bundles* podem ser agregados num só. Um *bundle* contém a hora a que foi criado, qual a sua duração, um designador de classe de serviço e o seu tamanho. Estas informações podem ser uteis na hora de agendar o seu envio e também na seleção do caminho mais indicado. O facto de se conhecer estas informações de antemão, possibilita que seja escolhido o caminho com melhores características e melhor desempenho para um determinado *bundle*. A origem e o destino de cada um dos nós é identificada através de *Enpoit Identifiers* (EIDs) [3].

Este protocolo tenta não só assegurar confiança nas transmissões de dados entre aplicações, como também tenta ser o mais versátil possível, tendo em conta as diferentes redes onde pode ser utilizado [13].

[2] Em português: "Unidade de dados da Aplicação"
[3] Em português: "Unidade de Dados do Protocolo"

2.3.1. Camada *Bundle* na Pilha Protocolar

Por forma a garantir a sua operacionalidade no interior e exterior de cada rede, independentemente do protocolo utilizado pela mesma, as DTNs implementam uma camada denominada de *Bundle Protocol*.

Tal como se pode verificar na Figura 3, a camada que implementa o *Bundle Protocol* situa-se logo abaixo da camada Aplicação. Esta disposição permite que este protocolo ofereça um serviço de transmissão fim-a-fim eficiente e de confiança [13].

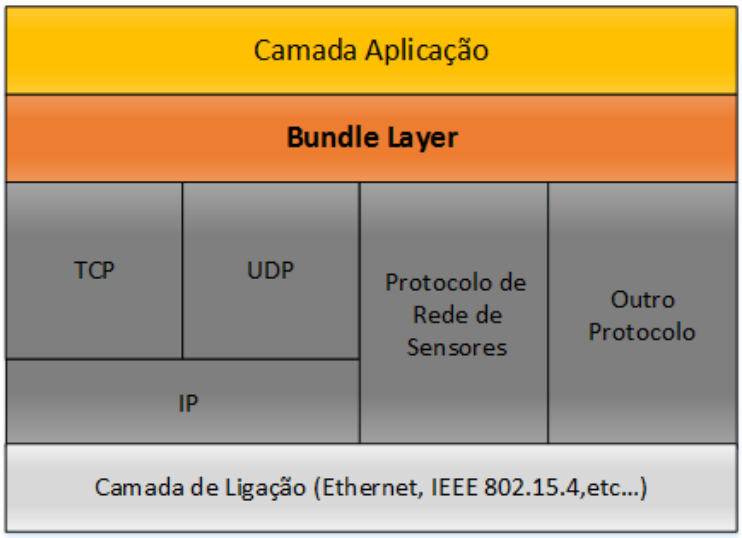


Figura 3 - Posição da *Bundle Layer* nas diferentes Pilhas Protocolares (adaptado de [12]**)**

O *Bundle Protocol* tem a capacidade de comunicar com os diferentes protocolos de transporte devido à Camada de Convergência. Esta é uma camada que está ligada à *Bundle Layer*. Existem várias adaptações da Camada de Convergência para os diferentes tipos de protocolo de transporte. Essas diferentes adaptações têm o nome de *Convergence Layer Adapters*[4] (CLAs). Existem atualmente várias CLAs já definidas, incluindo TCP, UDP, Bluetooth, entre outros [14].

[4] Em português: "Adaptações da Camada de Convergência"

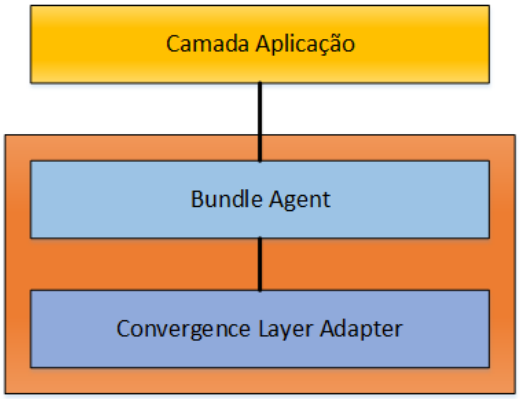


Figura 4 - Descriminação do conteúdo da *Bundle Layer*

Através da Figura 4 pode ver-se a "real" estrutura da *Bundle Layer*. É então possível distinguir-se duas subcamadas: o *Bundle Agent*, e a *Convergence Layer Adapter*. A primeira subcamada, não é mais que a implementação do *Bundle Protocol*. A segunda subcamada é a camada de convergência referida anteriormente. Nesta dissertação irá referir-se a *Bundle Layer* como sendo apenas uma camada, ou seja, a junção das duas subcamadas, não sendo feita assim a descriminação apresentada na Figura 4.

É nesta "nova" camada, implementada pelas DTNs acima das diferentes pilhas protocolares que tudo acontece. No nó de origem, a *Bundle Layer* recebe os dados das aplicações (*Aplication Data Units,* ADUs) e são agregadas as informações necessárias para possibilitar a entrega da mensagem. Forma-se assim um *bundle*, o qual é depois enviado para a camada de Transporte. Já no nó destino é efetuado o processo inverso. Ou seja, a camada de Transporte envia o *bundle* para a *Bundle Layer*. Nesta camada o *bundle* é desintegrado e apenas as informações uteis para a aplicação são enviadas para a camada superior. Nos nós intermediários, que apenas servem para fazer chegar a mensagem ao destino, não é efetuada qualquer alteração nos *bundles*, estes apenas são encaminhados. Na Figura 5 pode ver-se esquematicamente como funciona este processo.

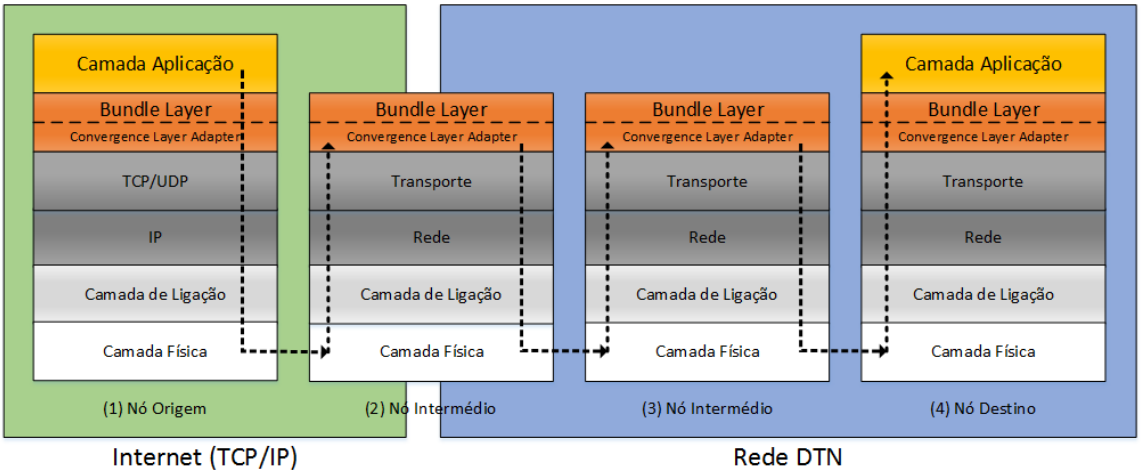


Figura 5 - *Bundle Protocol* (adaptado de [14]**)**

2.3.2. Estrutura *Bundle*

Como acontece em qualquer rede, também nas DTNs, os pacotes que circulam entre os nós devem obedecer a uma determinada estrutura. No caso das DTNs esses pacotes são os *bundles* e devem obedecer a uma estrutura especificada em [15].

Segundo a especificação a cima citada, um *bundle* deve ser a concatenação de pelo menos dois blocos. O primeiro deverá ser o bloco primário, sendo que, em cada *bundle* apenas pode existir um bloco deste tipo. O segundo bloco poderá ser o bloco de *payload*[5] no qual é transportada a informação útil, ou então algum outro bloco adicional, como por exemplo um bloco de segurança. Estes blocos adicionais são opcionais e denominam-se de "*extension blocks*" [15].

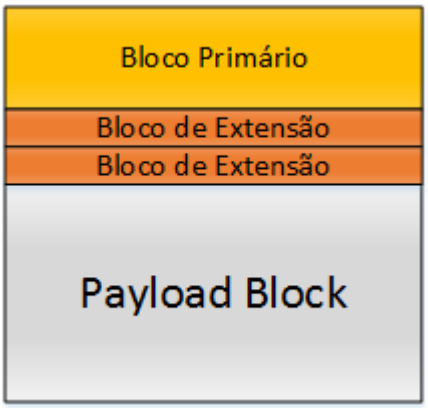


Figura 6 - Estrutura de um *bundle*

[5] Em português: "Carga; Dados;"

Através da Figura 6 pode ver-se o formato simplificado de um *bundle*. Tal como dito anteriormente existe o bloco primário, que é único e obrigatório em cada *bundle*; Existem os blocos de extensão que são opcionais, podendo existir ou não; E por fim existe o bloco *payload* no qual são transportados os ADUs, ou seja, a informação útil, que é proveniente da aplicação e importante para o utilizador.

Uma vez que o bloco primário é obrigatório em todos os *bundles* é nele que estão contidos os principais campos que caracterizam o *bundle*. Os campos presentes no bloco primário existem em todos os *bundles* e fragmentos [3]. Existem vários campos presentes no bloco primário, tendo cada um a sua função. Contudo, apenas os mais relevantes serão abordados.

Posto isto, os campos mais importantes presentes no bloco primário são: Tempo de Criação; Tempo de Vida; Classes de Serviço; EID Fonte; EID Destino; Reporta ao EID e EID de Custódia.

O campo "Tempo de Criação", juntamente com o EID do nó de origem, serve para identificar o *bundle*. Este campo é como que uma concatenação do instante em que determinado *bundle* foi criado e um número de sequência. Um *Bundle Protocol Agent* nunca deverá criar dois *bundles* distintos com o mesmo EID e o mesmo tempo de criação [3], [15].

O campo "Tempo de Vida" indica o momento em que a informação carregada num determinado *bundle* deixa de ser útil. Isto é, assim que o momento atual é maior que o tempo de criação mais o tempo de vida, então a informação contida no *bundle* deixa de ser útil. Sendo assim, os nós não necessitam mais de guardar ou encaminhar. O *bundle* deve ser eliminado da rede [3], [15].

No que diz respeito ao campo "Classes de Serviço" este indica as opções de entrega do *bundle*, incluindo a sua prioridade. Existem três tipos de prioridade: baixa, normal, expedita, sendo esta ultima a de maior prioridade. O conceito de prioridade será discutido mais adiante nesta dissertação.

O campo "EID Fonte" serve para identificar a origem do *bundle*. Tal como o campo "EID Destino" serve para identificar o destino do *bundle* [3].

"Reportar ao EID" é o campo que indica a quem é que devem ser endereçados os relatórios de entrega do *bundle*. Estes relatórios podem ser endereçados a qualquer nó da rede, podendo ser o nó de origem do *bundle* ou não [3].

Por fim, o campo "EID de Custódia" indica qual o nó que tem a custódia do *bundle* (isto se existir algum) [3]. O conceito de custódia será explorado e devidamente explanado mais adiante nesta dissertação.

O bloco de *payload* tal como já dito anteriormente transporta as ADUs, ou seja, a informação útil proveniente da aplicação e realmente importante para o utilizador.

Para além destes dois blocos, cada *bundle* pode conter ainda outros blocos, denominados de "Blocos de Extensão". São considerados de extensão todos os blocos que não são o bloco primário ou o bloco de *payload*. Uma vez que estes blocos não estão definidos em [15], nem todos os nós serão capazes de processar todos os blocos de extensão. Poderá acontecer o caso de um nó receber um *bundle* e não ser capaz de processar alguns ou todos os seus blocos de extensão [15].

Tal como dito anteriormente, o *Bundle Protocol* é um protocolo que opera acima das diferentes pilhas protocolares das diferentes redes. Sendo assim, quando um *bundle* é criado e é enviado para a camada abaixo da *Bundle Layer*, este fica sujeito aos protocolos que estão a operar nessa rede. Isto é, se um determinado nó DTN está a operar numa rede TCP/IP, por exemplo, o *bundle* criado nesse nó, fica então sujeito a esse protocolo de transporte.

Neste caso, os *bundles* criados são enviados para a camada de Transporte, na qual opera o protocolo TCP. Nesta camada são transformados em "segmentos TCP". Posteriormente estes segmentos são enviados para a camada de rede, na qual opera o protocolo IP, que converte os segmentos em datagramas IP. De seguida esses datagramas são enviados para a camada de ligação, na qual são convertidos em tramas. Por fim, as tramas são enviadas para a camada física, a qual as vê apenas como sendo bits que devem ser enviados.

Dependendo do protocolo que está a operar, em cada camada são inseridos cabeçalhos característicos desse protocolo. Esses cabeçalhos permitem aos protocolos oferecer serviços distintos. Os *bundles* assim que são enviados para a camada de transporte, são vistos como dados provenientes diretamente da aplicação e não têm qualquer tipo de tratamento especial por parte dos protocolos que estão a operar numa determinada rede. Desta forma, é garantida a versatilidade que tanto caracteriza as DTNs.

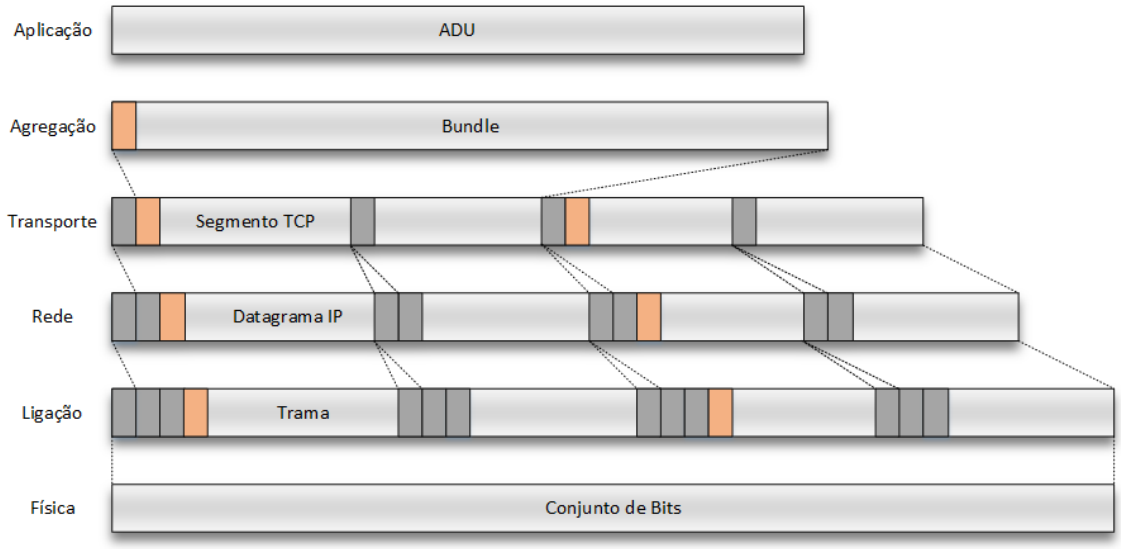


Figura 7 - Exemplo de Encapsulamento utilizando o protocolo TCP/IP (adaptado de [2]**)**

Na Figura 7 é dado um exemplo esquemático do encapsulamento de *bundles* utilizado na Internet atual. No exemplo da figura acima pode ver-se que o *bundle* que se pretende transmitir não cabe num segmento TCP apenas, pelo que é dividido em dois segmentos.

Durante todo este processo, um *bundle* pode ser "partido", isto é, fragmentado. Visto que um *bundle*, aos olhos da restante pilha protocolar, é visto como sendo dados provenientes da aplicação, não existe qualquer tipo de preocupação em envia-lo de uma só vez. Por exemplo, na Figura 7, um *bundle* pode ficar dividido entre dois segmentos. Sendo assim o *bundle* acaba por ser fragmentado. Se tal acontecer, o nó DTN de destino terá de voltar a juntar os segmentos num único *bundle* [2].

2.3.3. Fragmentação

Uma vez que as DTNs operam em ambientes onde as ligações são bastante intermitentes, foi necessário conceber uma estratégia que permitisse contornar esse obstáculo. Para tal, foi aplicado o conceito de "fragmentação". Este conceito permite que um *bundle* seja "partido" em dois ou mais fragmentos. Desta forma, o tamanho do *bundle* fica bastante reduzido permitindo que seja enviado de uma forma mais rápida [15].

Frequentemente, nos ambientes onde as DTNs operam e se destacam em relação aos outros tipos de rede, as ligações podem falhar a qualquer momento, ou podem ter uma velocidade extremamente baixa, podem ainda ser muito curtas ou então o *bundle* que se pretende enviar pode ser muito grande. No fundo, existe um limite tanto na quantidade de

dados que podem ser enviados, como no tempo em que uma ligação está disponível [14]. Consequentemente, para lidar com estas possíveis limitações é necessário recorrer à fragmentação. Este mecanismo aumenta a conectividade nas redes DTN, uma vez que permite explorar os contactos mais curtos [16]. Existem então dois tipos de fragmentação: Pró-ativa e reativa.

A fragmentação pró-ativa acontece quando o nó que envia o *bundle* decide *à priori*, ou seja, antes de o enviar, que este deve ser fragmentado. No fundo, um *bundle* é dividido em várias partes e cada uma delas é enviada como sendo um *bundle* independente [3]. Este tipo de fragmentação apenas é utilizado quando são conhecidas, ou é possível prever, as características da ligação (tempo disponível, velocidade, etc…). Só desta forma o nó que envia o *bundle* é capaz de decidir se este deve ser fragmentado ou não. Dá-se então a fragmentação quando o nó DTN decide que o *bundle* a ser enviado exige mais da ligação do que aquilo que esta pode fornecer [12]. É um mecanismo bastante parecido com o utilizado nas redes TCP/IP.

No caso da fragmentação reativa, esta dá-se *à posteriori*, ou seja, depois do *bundle* ser enviado. Este tipo de fragmentação acontece quando um *bundle* está a ser enviado e por algum motivo a ligação falha. Ou os nós DTN se afastam e a ligação perde-se, ou aparece algum obstáculo que interrompe a ligação, ou qualquer outro acontecimento que faça com que a ligação falhe. Este tipo de fragmentação é vista como uma reação à interrupção de uma ligação [12].

Nesta fase parte da informação foi já transmitida, parte ficou por transmitir, ou seja, o *bundle* foi parcialmente transferido. Posto isto, o nó DTN que estava a transmitir a informação converte a parte do *bundle* que não foi transmitida num novo *bundle*, podendo ser encaminhado assim que a ligação volte a estar disponível. Este novo *bundle* (fragmento), uma vez que é independente da restante informação já enviada, pode ser enviado para outro nó que não o nó para o qual havia sido enviada a outra parte da informação. Já o nó que estava a receber a informação converte o que foi recebido num novo *bundle*, acionando apenas o campo que indica que se trata de um fragmento e encaminha-o normalmente [3]. Este novo processo de fragmentação introduzido nas DTNs tem como objetivo evitar que seja reenviada informação que já foi corretamente enviada e recebida pelo outro nó [14]. Os novos fragmentos que surgem através deste processo são convertidos em *bundles* estando assim sujeitos a nova fragmentação se tal for necessário.

2.3.4. Nós, *Endpoints* e Registos

Tal como acontece nas redes TCP/IP, também nas DTNs existe o conceito de "nó". Posto isto, nas DTNs um nó é qualquer equipamento no qual opera o *Bundle Protocol*. Ou seja, um nó é visto como um qualquer dispositivo capaz de lidar com *bundles* [3]. As aplicações utilizam os nós DTN para receberem e enviarem *bundles*, os quais contêm informação necessária para o seu funcionamento, ADUs [3]. Os nós DTN estão para este tipo de redes como os *hosts* e os encaminhadores estão para as redes TCP/IP.

Sendo assim, um determinado nó pode a qualquer momento atuar como nó de origem ou de destino, ou como encaminhador. Os nós que atuam como nós de origem ou de destino são nós que enviam (nó origem) ou recebem (nó destino) *bundles*. Contudo, estes nós não encaminham informação vinda de um outro nó. Já um nó que atue como encaminhador pode operar sob duas situações distintas.

A primeira dá-se quando um nó encaminha informação para outros nós que possuem a mesma pilha protocolar. Relembrando o tópico acima abordado, estes nós possuem apenas uma CLA, isto é, utilizam apenas uma camada de convergência. Nesta situação, caso as ligações sejam lentas ou bastante intermitentes, os nós devem possuir armazenamento persistente. Desta forma, é garantido que os *bundles* não se perdem se por algum motivo a ligação não estiver disponível. Estes nós podem, opcionalmente, suportar transferência de custódia.

A segunda situação acontece quando o nó em questão se encontra na fronteira da rede. Neste caso, o nó funciona como um *gateway* e encaminha informação para outros nós que não têm a mesma pilha protocolar. Os nós que atuam nestas condições devem garantir que existe armazenamento persistente [2].

Contudo, ao contrário do que acontece nas redes TCP/IP, existe nas DTNs o conceito de "*endpoint*". Um *endpoint* não é mais que um conjunto de um ou mais nós, sendo que, um nó pode pertencer a mais que um *endpoint*. Existe ainda, nas redes DTN, o conceito de grupo mínimo de receção (*Minimum Reception Group*, MRG), que não é mais que um número mínimo de nós aos quais um *bundle* deve chegar para que a entrega seja considerada bem-sucedida [3]. Por outras palavras, MRG é um conjunto de nós, dentro de um *endpoint*, que devem receber determinado *bundle*. Se tal não acontecer, então a entrega não foi efetuada com sucesso.

A MRG define ainda a semântica da comunicação. Isto é, dependendo do número de nós que compõem MRG a comunicação é *unicast*, *multicast*, *anycast* ou *broadcast*. Se a MRG for constituída apenas por um nó a comunicação é *unicast*. Já se a MRG for apenas um nó por entre

um grupo deles a comunicação é *anycast*. Por fim, se a MRG for constituída por todos os nós de um grupo a comunicação é *multicast* ou *broadcast*, se aquele for o único grupo [3].

Por forma a identificar cada um dos *endpoints* existentes numa rede, é dado a cada um deles um nome. Nas DTNs os "nomes" que são atribuídos aos *endpoints* são denominados de "*Endpoint Identifier*" (EID). Tal como referido anteriormente, um *endpoint* é um conjunto de um ou mais nós. Sendo assim, um EID pode referir-se a apenas um nó, ou então, pode referir-se a um conjunto deles (*multicast*). Neste caso, todos os nós são identificados pelo mesmo EID. O caso em que um EID se refere a apenas um nó, é denominado de "*singleton endpoint*". Desta feita, numa comunicação, os nós que originam a mensagem são *sigleton endpoints*, já o destino pode também sê-lo ou não, dependendo se o destino é apenas um nó ou um conjunto deles (*multicast*, conjunto de nós com o mesmo EID) [2].

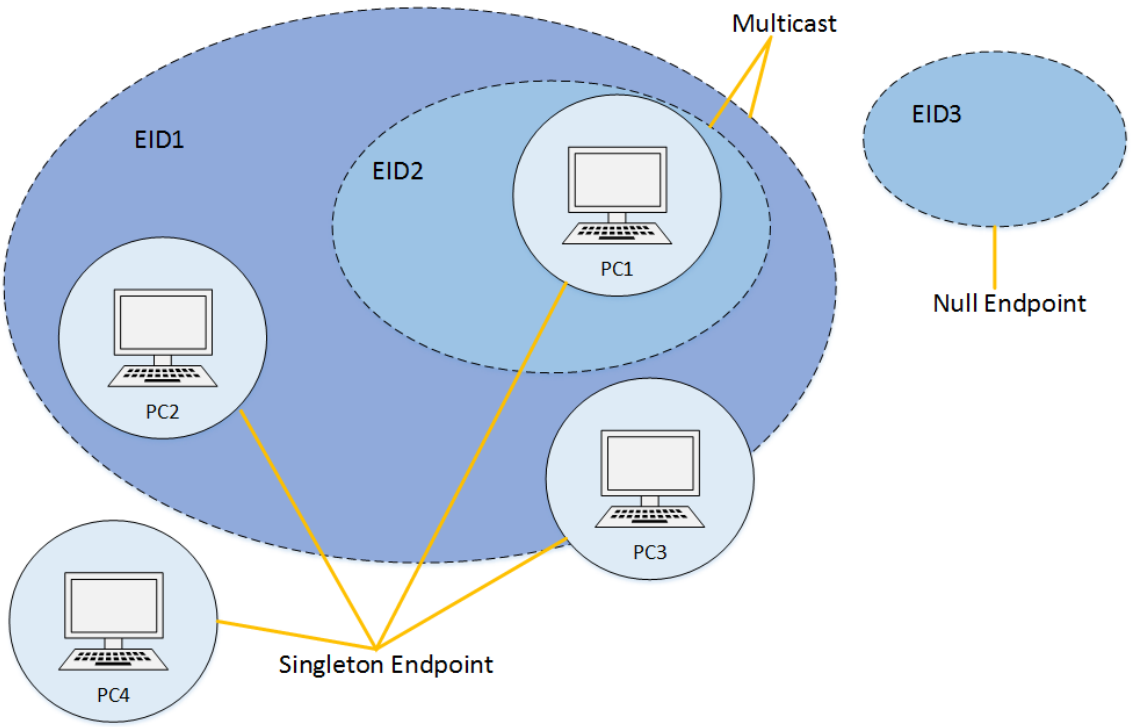


Figura 8 - Exemplo de Distribuição de EIDs (adaptado de [2])

Na Figura 8 pode ver-se a distribuição de EIDs numa rede. Cada um dos terminais (PC1, PC2, PC3 e PC4) possui o seu próprio EID, sendo que estes são *Singleton Endpoint* uma vez que se referem a apenas um nó. Já um determinado *bundle* que tenha como destino o EID1, deverá chegar aos nós PC1, PC2 e PC3 (*Multicast*), dado que todos estes três terminais estão também identificados com este EID. Por fim, se existir um *bundle* com destino EID3, este não será

entregue a ninguém, uma vez que não existe qualquer nó que seja identificado por esse EID. O EID3 é um *Null Endpoint*.

Este conceito de EID é bastante importante nas DTNs uma vez que quando produzem informação, as aplicações endereçam as ADUs a um determinado EID. Por outro lado, no *Bundle Protocol* não existe o conceito de endereço, pelo que as estratégias de roteamento são baseadas nos EIDs [14]. Posto isto, o EID deve conter por si próprio informação que permita o encaminhamento do *bundle*.

Um EID é expresso sintaticamente como sendo um "*Uniform Resource Identifier*" (URI) [17]. Este esquema de nomeação foi desenvolvido para expressar nomes ou endereços com uma vasta gama de situações. Por este facto é uma boa estratégia para utilizar nos nomes dos DTN *endpoints* [3].

Nas DTNs o URI utilizado para identificar os *endpoints* tem um formato dividido em duas partes: a primeira parte é denominada de "*scheme_name*" [6] e a segunda de "*scheme-specific_part*"[7] (SSP). Um EID tem então o seguinte formato:

"**<scheme_name> : <scheme-specific_part>**"

Nas DTNs a primeira parte da *string* é, por defeito, "dtn". Já a segunda parte pode ser bastante geral, uma vez que representa um nome que deverá ser resolvido internamente assim que o *bundle* chegar a uma determinada rede [1]. Ou seja, a segunda parte do EID utilizando esta sintaxe URI é como que a definição do nó de destino. Assim que o *bundle* chega à rede de destino, esta segunda parte é então manipulada de acordo com os protocolos que operam nessa mesma rede. O objetivo é determinar qual o nó de destino, ou no caso de uma comunicação *multicast*, quais os nós finais. Por exemplo, numa rede TCP/IP um EID teria o seguinte formato [14]:

"**dtn://dtn.example.com/myApp**"

No exemplo acima, o ultimo passo para que a informação chegasse ao seu destino final passaria por verificar o endereço IP do nome "dtn.example.com" e depois encaminhar a informação para esse *host*.

[6] Em português: "Esquema do Nome"
[7] Em português: "Parte Especifica do Esquema"

A este mecanismo de determinação do nó final (ou o grupo de nós final) apenas quando o *bundle* chega à rede de destino dá-se o nome de "*Late Binding*"[8]. Este mecanismo significa que a conversão de um nome para um endereço não acontece necessariamente no nó de origem, mas sim no último *gateway*. Nas redes TCP/IP, antes de a informação ser enviada é consultado um servidor DNS, que contém como que uma lista onde está descriminado qual o endereço associado a determinado nome. Nas DTN isto não acontece. Os *bundles* são enviados e assim que chegam à fronteira da rede de destino, a segunda parte do nome, isto é, o SSP é localmente interpretado e convertido, se necessário, de acordo com o esquema de nomeação a operar na rede. Desta forma, é encontrado o endereço do nó de destino. Este mecanismo garante mais uma vez a versatilidade das DTNs, uma vez que, permite que qualquer rede utilize o seu próprio esquema de nomeação [1], [3].

Este mecanismo apresenta várias vantagens quando aplicado em redes onde as quebras de ligação são frequentes: reduz a quantidade de informação administrativa a circular na rede. Permite que novas redes, com novos esquemas de endereçamento próprios, sejam adicionadas sem que haja impacto nas restantes redes e nos nós já existentes [13]. Facilita a entrega da informação no seu destino final em casos de alteração na topologia da rede que ainda não tenham sido detetadas pelo nó de origem [12]. Um determinado *bundle* pode demorar bastante tempo a chegar ao seu destino final podendo exceder o período de validade de um vínculo, tornando desta forma impossível ou inválido fazer esta ligação entre o EID e o endereço do destino final no nó de origem.

Para além de tudo isso, este mecanismo é vantajoso se no ambiente onde opera a rede DTN alguns dos nós não conseguirem aceder a determinadas infraestruturas como os servidores de DNS nas redes TCP/IP, por exemplo [14].

Por fim, o facto de não necessitar de um servidor físico (servidor DNS nas redes TCP/IP, por exemplo) para estabelecer esta associação entre o nome e o endereço de determinado nó reduz o atraso na entrega dos *bundles*, isto porque, caso existisse essa necessidade, o servidor poderia encontrar-se numa outra rede longe do nó de origem o que numa rede caracterizada por atrasos nas comunicações seria impraticável [1].

Existe também nas DTNs o conceito de "Registo". Um registo acontece quando uma determinada aplicação demonstra interesse em receber ADUs destinadas a um determinado EID. Esse registo é, normalmente, guardado na memória persistente de um nó DTN, desta forma, as informações do registo e da aplicação em si são capazes de sobreviver a uma paragem

[8] Em português: "Vinculo Tardio"

inesperada e consequente reinicio da própria aplicação ou do sistema operativo. Sendo assim, se tal acontecer, não é necessário que seja feito outro registo [3].

2.3.5. Transferência de Custódia

No protocolo TCP/IP utilizado atualmente na Internet, existem mecanismo que permitem confiabilidade fim-a-fim. Ou seja, caso um determinado pacote chegue ao seu destino com erros, ou com algum tipo de corrupção, ou não chegue de todo, existe a possibilidade de pedir a sua retransmissão. A aplicação que recebe o pacote encontra algum tipo de erro na informação recebida e pede então que essa mesma informação seja reenviada pela fonte. Neste tipo de redes esta abordagem não é problemática, uma vez que os nós se encontram sempre disponíveis [18].

Contudo, nas DTNs tal não é viável. Isto porque, devido à grande instabilidade das ligações entre os nós, um nó fonte pode deixar de estar disponível a qualquer momento. O exemplo utilizado anteriormente, das sondas no oceano que pretendem enviar informações para terra, é um bom exemplo de uma situação onde tal pode ocorrer. Como se sabe, uma sonda tem recursos de energia e armazenamento limitados. Sendo assim, caso seja necessário proceder ao reenvio das informações, estas podem já não se encontrar disponíveis. A sonda pode, por exemplo, ter-se desligado devido à falta de energia, ou pode então não ter capacidade de memória para guardar informação durante todo esse tempo.

Para garantir algum tipo de fiabilidade nas comunicações, o *Bundle Protocol* introduz um mecanismo denominado de "Transferência de Custódia". Neste mecanismo, são os nós originadores que têm a responsabilidade de reenviar o *bundle*, caso seja necessário. Existem mecanismos que permitem a transferência de custódia para outros nós no percurso para o destino. Desta forma garante-se que se determinado *bundle* que chega corrompido ao destino possa ser reenviado, mesmo que o nó que o originou já não se encontre conectado à rede por algum motivo [14]. Este mecanismo garante confiabilidade na transferência, em que a responsabilidade vai sendo transferida para nós que se encontram mais adiante no percurso. Observe-se que esta transferência não precisa de ser feita nó a nó. Suponhamos, por exemplo, que um *bundle* é originado num nó "A" e ao longo do seu trajeto passa pelos nós "B", "C", "D", "E" e "F". Inicialmente, a custódia é do no "A" e pode ser transferida para o nó "C" e depois para o nó "E".

Existe assim uma transferência de custódia, isto é, de responsabilidade, entre nós para que seja assegurada a retransmissão de um *bundle*. Contudo, nem todos os nós podem assegurar a

custódia de um *bundle*. Quando se efetua a transferência de custódia de um nó para outro, o nó que a vai receber deve verificar alguns requisitos [18]: deve estar mais perto do destino do *bundle* que o nó que possui atualmente a custódia; terá ainda de garantir ser capaz de guardar o *bundle* por um longo período de tempo; deve ser capaz e ter vontade de entregar o *bundle* no seu destino final; garantir que tem energia suficiente para ficar ativo por um longo período de tempo; e por fim, deve garantir que aproveita todas as chances que tiver para encaminhar o *bundle*.

Posto isto, não é garantido que todos os nós devam aceitar a custódia de um determinado *bundle*. Isto porque, numa determinada fase um nó pode ter memória suficiente para alojar permanentemente o *bundle*. Contudo, num outro momento em que se encontre mais congestionado ou com pouca energia o nó recusará a custódia [3].

Este mecanismo opera de uma forma bastante simples. No nó de origem, quando um *bundle* é criado, a aplicação pode requerer que os nós ao longo do caminho assumam a custódia desse mesmo *bundle*. Assim sendo, um nó "A" que detém a custódia de um *bundle* envia-o para o próximo nó, o nó "B", seguido de um pedido especial para que este assuma a custódia desse mesmo *bundle*. Após o envio da informação e do pedido de transferência de custódia, o nó A ativa um contador. O nó B, por sua vez, deve enviar um ACK para o nó A, confirmando a transferência de custódia. Caso o nó B não envie essa confirmação, o nó A espera que o contador chegue ao fim. Assim que se dá o *timeout*, o nó A reenvia o *bundle* e um novo pedido, sendo que, este processo repete-se até que a confirmação chegue. Assim que a confirmação da transferência de custódia chega ao nó A, este apaga o *bundle* da sua memória. Confirmada a transferência de custódia, o nó B é agora o responsável por reenviar o *bundle* caso assim se justifique. Ao longo do caminho, todo este processo é repetido até que a informação chegue ao seu destino final.

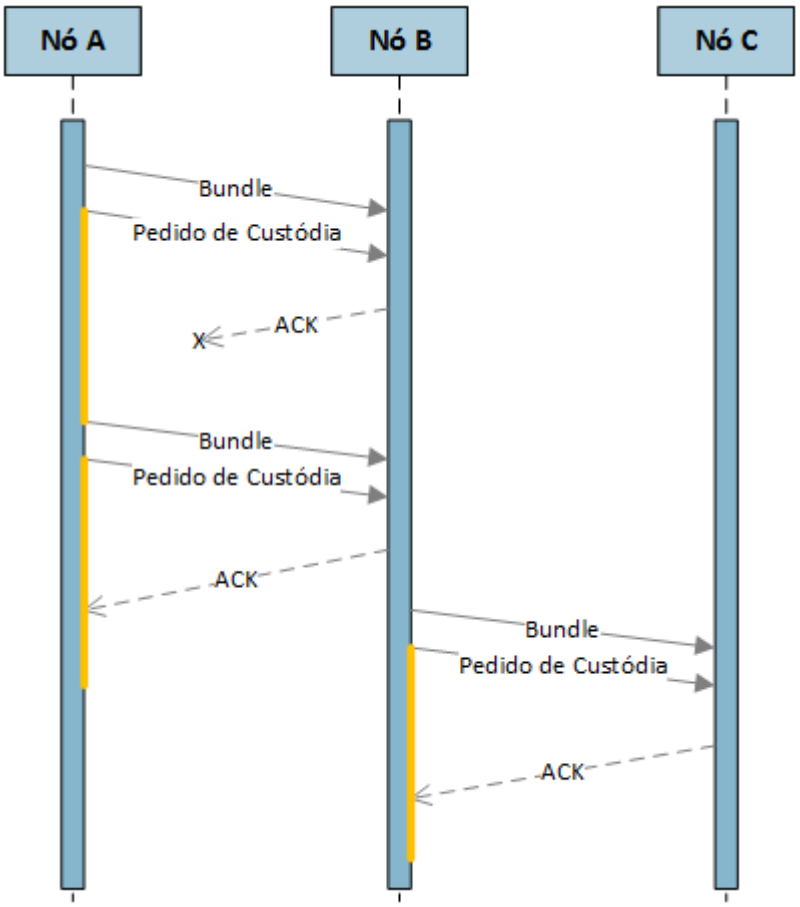


Figura 9 - Envio de um *bundle* e Pedido de Transferência de Custódia

Através da Figura 9 pode ver-se o processo de transferência de custódia entre três nós A, B e C. Na primeira situação o nó A envia o *bundle* para o nó B, seguido de um pedido de custódia. De seguida, o nó A inicia o seu contador (amarelo). O nó B aceita o pedido de custódia e envia o ACK para o nó A. Contudo, o ACK proveniente do nó B perde-se, por algum motivo, e não chega ao nó A. Assim que o contador do nó A termina, este volta a enviar o *bundle* seguindo de um novo pedido de custódia. O nó B aceita mais uma vez a custódia desse *bundle* e envia o ACK para A. Desta vez, o ACK não se perde e chega corretamente ao nó A. Fica assim completa a transferência de custódia do nó A para o nó B. O processo repete-se quando o *bundle* é encaminhado do nó B para o nó C, contudo, desta vez tudo corre bem à primeira.

Um nó que assuma a custódia de um qualquer *bundle* deverá manter uma cópia deste na sua memória persistente até que um outro nó assuma a custódia ou então até que o seu tempo de vida seja ultrapassado [2].

2.3.6. Contactos

Nas DTNs, como referido anteriormente, as ligações entre os nós são muito instáveis, podem nem sequer existir ou podem ainda ter um atraso muito elevado ou uma velocidade extremamente baixa. Contudo, há momentos em que existem efetivamente ligações, minimamente estáveis, que permitem que a informação avance em direção ao seu destino final. A essas ligações dá-se o nome de "contactos".

Posto isto, um contacto é tido como sendo um período de tempo no qual dois nós comunicam entre si e são capazes de trocar informação. Durante este período de tempo, o atraso e a velocidade comunicação são considerados constantes. Existe ainda o conceito de "volume", que não é mais que o produto entre a capacidade e o tempo que demorou esse contacto. Isto é, o volume pode ser visto como a quantidade de informação que foi transferida nesse contacto [18]. Caso seja conhecido o momento em que se dá um determinado contacto e o seu volume, podem ser tomadas melhores decisões de encaminhamento e pode também influenciar na escolha da próxima mensagem a ser enviada, aumentando assim a velocidade de entrega [3], [1].

Existem então cinco tipos de contacto: Persistentes; Sob Demanda; Agendados; Oportunistas e Previsíveis.

Os contactos "Persistentes" estão sempre disponíveis, isto é, são ligações que estão sempre ativas e podem sempre ser utilizadas para trocar informação.

O tipo de contactos "Sob Demanda" são contactos que exigem ser ativados, ou seja, são contactos que necessitam de um qualquer tipo de ação para que se encontrem disponíveis. Após estarem disponíveis estes contactos comportam-se como contactos persistentes até que sejam terminados.

No tipo de contacto "Agendados" é necessária a existência de um acordo entre as duas partes. Nesse acordo é definido o momento em que o contacto deve acontecer e também a sua duração. Neste tipo de contactos é requerido alguma sincronização entre ambas as partes para que o contacto aconteça no tempo previsto. Caso seja marcado para determinado momento num dos nós, é necessário que no outro seja marcado para o mesmo espaço temporal [2].

Os contactos "Oportunistas" acontecem espontaneamente, sem que sejam previstos ou anteriormente arranjados. Neste tipo de contactos também a duração não é conhecida, podendo terminar assim que a troca de informação ficar completa ou pode também terminar por qualquer outro motivo, deixando a troca de informação a meio.

Por fim, nos contactos “Previsíveis” também não é conhecido quando acontecem, contudo, espera-se que aconteçam em determinado momento e que durem no mínimo um determinado período de tempo. Esta expectativa é baseada na observação de contactos que aconteceram anteriormente e no seu comportamento.

Posto isto, é então compreensível que os contactos e as suas características são fundamentais para que a informação chegue ao seu destino. De algum modo, os contactos influenciam a forma como a informação é encaminhada e são determinantes na escolha do protocolo de encaminhamento que deve ser utilizado em determinada rede [14].

2.4. Encaminhamento nas DTNs

2.4.1. Classes de Prioridade

Tal como o tipo de contacto, também a prioridade de entrega influencia a forma como a informação é encaminhada. A DTN permite que existam três tipos de prioridade: baixa, média e alta. Esta distinção é feita pelas aplicações que geram a informação (ADUs). As aplicações requerem uma classe de prioridade para todos os ADUs que criam. Sendo assim, uma aplicação pode requerer urgência máxima na entrega da informação acabada de criar, pelo que, o *bundle* onde essa informação é encapsulada terá prioridade alta aquando do seu encaminhamento até ao destino final.

Esta definição de prioridade apenas serve para priorizar *bundles* que tenham a mesma origem. Isto é, se dois *bundles* têm a mesma fonte, esta é uma forma de distinguir qual deles deve ser enviado primeiro. Sendo assim, um *bundle* marcado como tendo uma prioridade alta, pode ser encaminhado depois de um outro *bundle* com outra origem marcado com prioridade média. Esta gestão de qual dos dois *bundles* deve ser enviado primeiro é feita pelo nó DTN tendo em conta vários fatores, como por exemplo o seu destino, o tempo de vida de informação ou o protocolo de encaminhamento que está a ser utilizado [3].

Posto isto, existem então três distinções de prioridade: Baixa, Normal e Expedita.

Os *bundles* marcados como baixa prioridade são os últimos a serem encaminhados, sendo que esta é a prioridade mais baixa. Estes *bundles* são apenas enviados após todos os outros *bundles* com outra prioridade e com o mesmo endereço de origem e de destino terem sido enviados [3].

Já os *bundles* marcados como *normal* são enviados antes de qualquer outro *bundle* marcado como sendo de baixa prioridade. Posto isto, a classe de prioridade *normal* é a segunda a ser escolhida no processo de encaminhamento até ao destino final.

Por fim, a classe de prioridade expedita surge como sendo a classe de maior prioridade, pelo que, os *bundles* marcados com esta prioridade são os primeiros a serem reencaminhados, antes de qualquer outro *bundle* com qualquer outra prioridade.

2.4.2. Protocolos de Encaminhamento

Tal como em outros tipos de redes, também nas redes tolerantes a atrasos existem protocolos de encaminhamento, nomeadamente quatro. São eles: Entrega Direta, Epidémico, Spray-and-Wait e PRoPHET.

2.4.2.1. Entrega Direta

O protocolo Entrega Direta não utiliza encaminhamento. Neste protocolo, o nó que cria a mensagem, apenas entrega a mesma ao nó destino. Se o nó origem e o nó destino não se cruzarem, a mensagem não é entregue. Neste protocolo, não são utilizados outros nós para fazer com que a mensagem chegue ao destino. Apesar de praticamente não serem gastos recursos, as suas taxas de entrega são bastante baixas.

2.4.2.2. Epidémico

Encaminhamento Epidémico [19] é um protocolo de encaminhamento é baseado na ideia de inundar a rede com cópias da mensagem. Ou seja, uma cópia da mensagem é transmitida a todos os nós que ainda não a possuam. A cada novo contacto os nós trocam vetores. Esses vetores indicam quais as mensagens que determinado nó possui no seu *buffer* nesse momento. Após a análise desses vetores, são trocadas cópias de mensagens que não constem no *buffer* do outro nó.

Por exemplo, um determinado nó A encontra um nó B. Estes trocam vetores, indicando quais as mensagens que estão alojadas no seu *buffer*. O nó A, ao analisar o vetor do nó B, deteta que existem mensagens que estão no seu *buffer*, mas não no buffer do nó B. Sendo assim, procede à transmissão de cópias das mensagens em falta. O mesmo acontece com o nó B. Através da análise do vetor do nó A, deteta que existem mensagens que estão na sua posse,

mas não na posse do nó A. Assim sendo, é enviada uma cópia de cada mensagem do nó B para o A.

Este protocolo é particularmente vantajoso quando não são conhecidas informações acerca da rede. Contudo, o facto de inundar a rede com cópias de mensagens leva a que haja um consumo excessivo de recursos. Isto na medida em que a maioria dos nós terá uma cópia da mensagem, porém, não se cruza com o nó destino. Apesar desta desvantagem, este protocolo apresenta taxas de entrega bastante elevadas. Quando maior for o número de cópias de uma mensagem a circular na rede, maior será a rapidez com que essa mensagem é entregue.

2.4.2.3. Spray-and-Wait

Spray-and-Wait [20] é um protocolo que tenta unificar duas aproximações distintas, usufruindo dos benefícios de cada uma delas. Isto é, tenta beneficiar da alta taxa de entrega utilizando uma aproximação baseada na reprodução de pacotes, mas, simultaneamente, tenta usufruir da excelente eficiência na utilização de recursos característica de uma aproximação baseada no encaminhamento de pacotes.

Com o objetivo de aumentar a eficiência na utilização de recursos da rede, evitando assim que se utilizem recursos desnecessariamente, o protocolo Spray-and-Wait permite que determinado pacote seja reproduzido determinado número de vezes. Existem assim, um limite máximo do número de cópias de um pacote que podem circular na rede.

Este protocolo é composto por duas fases. A primeira fase denomina-se de "*spray phase*". Quando uma mensagem é criada, um número 'L' é anexado à mensagem ('L' é o número máximo de cópias daquela mensagem permitidas na rede). O criador da mensagem entrega uma cópia da mesma aos 'L' primeiros nós que encontrar. Assim que um nó retransmissor recebe a mensagem (isto é, uma cópia) entra na segunda fase do protocolo: a fase de espera. Nesta fase, os nós retransmissores mantêm a cópia da mensagem até que se cruzem com o destino da mensagem, procedendo assim à sua entrega direta, sem recorrer a outros nós.

2.4.2.4. PRoPHET

PRoPHET [10] é um acrónimo do inglês *"Probabilistic Routing Protocol using History of Encounters and Transmitivity"*. Este é um protocolo que tenta explorar a não casualidade no movimento dos nós no mundo real. Para tal, é utilizado um algoritmo que calcula a probabilidade de determinado nó se encontrar com o nó destino.

O funcionamento deste protocolo baseia-se na premissa de que se dois nós se encontram num determinado instante, têm maior probabilidade de se voltar a encontrar futuramente. Baseado no número de vezes que dois nós se contactam, o algoritmo acima referido calcula a probabilidade de se voltarem a encontrar. Cada vez que dois nós se encontram essa probabilidade aumenta. A cada contacto feito a probabilidade é recalculada e atualizada.

Por exemplo, se um determinado nó A encontra um nó B a probabilidade de se voltarem a encontrar novamente aumenta. Sendo assim, através de uma fórmula, a probabilidade de A se encontrar com B é recalculada e atualizada.

Quando uma mensagem é criada, o nó que a origina guarda uma cópia enquanto existir espaço disponível no seu *buffer*. O nó de origem só encaminha a mensagem para um nó que tenha uma maior probabilidade de encontrar o nó destino. Assim acontece com os restantes nós retransmissores. A cada contacto, a mensagem deve ser encaminhada para o nó que tem maior probabilidade de encontrar o nó de destino. Ou seja, voltando ao exemplo anterior com os nós A e B, se o nó B tem uma maior probabilidade de encontrar o nó de destino, então A deve encaminhar a mensagem para B por forma a atingir o destino mais rapidamente. No entanto, se o nó A tem uma maior probabilidade de se encontrar com o nó destino que o nó B a mensagem não é encaminhada.

As probabilidades vão diminuído à medida que o tempo passa e dois nós não se encontram. Se dois nós não se encontram durante muito tempo, é menos provável que se voltem a encontrar. Posto isto, existe neste protocolo uma constante de envelhecimento. Esta constante é aplicada numa equação que atualiza as probabilidades a cada novo encontro.

Este protocolo conta ainda com uma propriedade transitiva. Esta propriedade indica que se dois nós se encontram com frequência, e um deles tem uma grande probabilidade de encontrar um terceiro nó, então esse terceiro nó provavelmente é um bom retransmissor para mensagens destinadas ao primeiro nó. Isto é, se um nó A encontra com frequência um nó B e este nó B encontra com frequência um nó C, então, provavelmente, este último nó, é um bom nó para encaminhar mensagens destinadas ao nó A.

Capítulo 3 – Rede de Dados Nomeados

3.1. Introdução às NDNs

Tal como já dito anteriormente, a Internet é o meio de comunicação e de distribuição de informação mais utilizado. Com o aumento de equipamentos capazes de aceder a esta rede, a informação contida na Internet aumentou exponencialmente. Por outro lado, também as propriedades dos equipamentos que a ela acedem foram-se alterando ao longo dos anos desde que a Internet foi criada. Hoje em dia, a capacidade de armazenamento, por exemplo, é enorme quando comparado com os primeiros equipamentos a aceder à rede. Outra característica é a mobilidade desses mesmos equipamentos, sendo que, hoje em dia, a maioria dos equipamentos que acedem à rede são equipamentos moveis (*smartphones*, *tablets*, computadores portáteis, etc…). Um outro fator que se vem alterando nos últimos anos é a própria utilização da Internet. Isto é, com o aumento do *e-commerce*[9], das redes sociais e das aplicações para *smartphones* a Internet tem sido cada vez mais vista como sendo um meio de distribuição de conteúdos do que propriamente um meio de comunicação entre duas máquinas [5].

Também como referido anteriormente, a arquitetura da Internet é baseada no protocolo TCP/IP. Este protocolo é baseado nos nós, ou seja, toda a rede opera em torno dos nós, das ligações existentes entre eles e da sua posição. Sendo assim, este protocolo foi desenvolvido para criar uma rede de comunicação (fim-a-fim), e tem dificuldades em operar como uma rede de distribuição de informação [5]. Para uma rede desenvolvida para conversas fim-a-fim é bastante complicado tornar-se numa rede de distribuição, tal como as necessidades de hoje em dia requerem. Por exemplo, uma rede telefónica, desenhada para conversas ponto-a-ponto seria um péssimo veículo para uma transmissão *broadcast* como televisão ou rádio [6]. O facto de se utilizar uma rede de comunicação como uma rede de distribuição é bastante complexo e não muito eficiente. Convém não esquecer que a Internet foi criada tendo como base a rede telefónica.

[9] Em português: Comércio Online

Vejamos um simples exemplo: um estafeta que distribui jornais. Com a abordagem utilizada nos dias de hoje (TCP/IP) o estafeta parte da redação com um jornal e leva a casa do leitor. Entretanto, um outro leitor, vizinho do primeiro, manifesta o seu interesse em ler o jornal. Sendo assim, o estafeta volta para a redação onde pega noutro jornal e leva a casa deste outro leitor. Facilmente se verifica que existe um enorme desperdício de recursos. Já se for utilizada uma outra abordagem, o estafeta leva várias cópias do jornal logo da primeira vez e entrega-os nas casas dos respetivos leitores. Desta forma, existe uma grande poupança de recursos quando comparada com a primeira abordagem. Na arquitetura da Internet atual passa-se praticamente o mesmo, excetuando o facto de não se levarem vários pacotes de uma vez (fazem-se sim várias cópias do mesmo pacote assim que existirem interesses iguais vindos de nós distintos). Uma informação chega a um nó destino, contudo, se o seu nó vizinho estiver interessado nessa mesma informação, terá de fazer um novo pedido e esperar que a informação chegue, desperdiçando deste modo recursos em largura de banda.

Posto isto, tem surgido nos últimos anos uma nova abordagem: Redes de Dados Nomeados (*Named Data Network*, NDN). Esta rede propõem uma abordagem diferente na forma como os pacotes que contêm a informação são tratados. É sugerido então que se transite de um conceito de "entregar este pacote a um determinado destino" para um conceito de "receber informação com determinado nome". Esta nova abordagem sugere que a forma como os pacotes são reencaminhados ao longo da rede deve ter em conta a informação que está contida em cada um dos pacotes e não nas posições do nó de origem e destino. Esta nova abordagem possibilita várias oportunidades [6]:

- Atualmente as aplicações são projetadas tendo em conta **qual** a informação que querem e não **onde** esta se encontra. Por esse facto é utilizado um *middleware* para converter a abordagem das aplicações na abordagem da Internet, baseada na localização e no endereço dos *hosts* onde se encontra a informação. Com as NDNs o modelo implementado pelas aplicações pode ser diretamente aplicado à rede, sem que seja necessário qualquer tipo de *middleware*.
- Uma vez que as conversações entre dois nós podem ser acerca de qualquer assunto, a abordagem de segurança utilizada hoje em dia para blindar um canal (de que se funciona bem para um, funciona bem para todos) pode não ser suficiente para preencher as necessidades de segurança fim-a-fim na comunicação entre um produtor e um consumidor. Nas NDNs toda a informação está segura fim-a-fim e o nome fornece ainda mais segurança. Com as NDNs é

possível saber se toda a informação que está a ser vista pelo consumidor foi produzida e assinada por um produtor legítimo.

- Sendo que todos os pacotes de dados podem ser nomeados unicamente, ou seja, apenas um determinado pacote tem aquele nome, pode prevenir-se situações de *looping* utilizando a memória dos encaminhadores.

É aplicado neste tipo de redes o conceito de publica/subescreve. Ou seja, existem nós que produzem determinado tipo de informação e anunciam à rede que são produtores dessa informação. Existem também nós que estão interessados num determinado tipo de informação e, sendo assim, subescrevem essa informação. Desta forma, é possível que dois nós recebam a mesma informação, proveniente do mesmo produtor, sem que este tenha que enviar a informação para cada um dos nós separadamente. Fica assim satisfeita a necessidade proveniente da Internet ser um meio de distribuição de informação.

As NDNs são uma das arquiteturas propostas existentes para a Futura Internet, designadas genericamente como Redes Centradas na Informação (ICN - *Information Centric Network*). Existem ainda outras propostas, como por exemplo: DONA (*Data-Oriented Network Architecture*, [21]), XIA (*eXpressive Internet Architecture*, [22]), PURSUIT (*Publish-Subscribe Internet Technologies*, [23]), entre outras.

Existem vários exemplos onde a utilização de um conceito deste tipo seria o mais adequado. A rede social Facebook é um bom exemplo. Devido às nossas relações pessoais e aos gostos parecidos que temos com os nossos amigos, seguimos mais ou menos as mesmas celebridades, temos um círculo de amigos bastante semelhante, acabamos por ver os mesmos conteúdos. Ou seja, é normal dois amigos fisicamente perto um do outro verem o mesmo conteúdo (fotos, posts, vídeos, etc…) em dispositivos diferentes. Desta feita, uma vez que dois equipamentos estão interessados no mesmo conteúdo, não haveria necessidade de fazer dois pedidos iguais ao produtor da informação. Bastava um deles fazer esse pedido e a informação era replicada.

Outro exemplo são os vídeos. Hoje em dia, existe o conceito de “vídeos virais” em que o mesmo vídeo é visto por milhões de pessoas em todo o mundo. Ou seja, é cada vez mais normal as pessoas verem os mesmos conteúdos. Este é mais um exemplo que na atualidade a Internet é vista como um meio de distribuição de conteúdos, em que é enviada informação para a rede e várias pessoas acedem à mesma.

3.2. Arquitetura das NDNs

Uma das características da arquitetura da Internet usada hoje em dia é o facto de se basear no princípio de ampulheta. Isto é, na atual arquitetura da Internet existe apenas uma parte comum a todos os nós, que é a parte mais estreita da ampulheta. Nessa parte mais estreita situa-se o protocolo IP e os pacotes baseados nesse protocolo. Acima da parte mais estreita situa-se o protocolo de transporte e a aplicação. Abaixo dela existe a tecnologia da rede utilizada. As características da parte mais larga podem ser diferentes em cada nó. Por exemplo, um nó pode receber pacotes IP através de um cabo de rede, outro pode receber pacotes através de uma tecnologia *wireless* (*wi-fi*, *Bluetooth*, entre outros). Independentemente do modo como recebem os pacotes, todos os nós que fazem parte da Internet têm em comum a tal zona mais estreita da ampulheta que lhes permite lidar com os pacotes IP. Acima da zona mais estreita situam-se os protocolos de segurança e as aplicações. Mais uma vez, esta zona pode ser diferente de nó para nó. Um nó pode utilizar os pacotes IP para satisfazer necessidades de uma determinada aplicação, tendo os seus protocolos de segurança. Por outro lado, um outro nó utiliza os pacotes IP para satisfazer necessidades de uma outra aplicação, sem qualquer segurança, por exemplo. A parte acima da zona estreita da ampulheta pode também ela ser diferente de nó para nó.

É na parte mais estreita da ampulheta (comum a todos os nós) que estão implementadas as funcionalidades mínimas que permitem a intercomunicação. Foi este princípio de ampulheta que ditou o sucesso da arquitetura atual. Isto porque permitiu que nas camadas acima e abaixo da zona estreita, se fossem desenvolvendo tecnologias sem qualquer limitação [6].

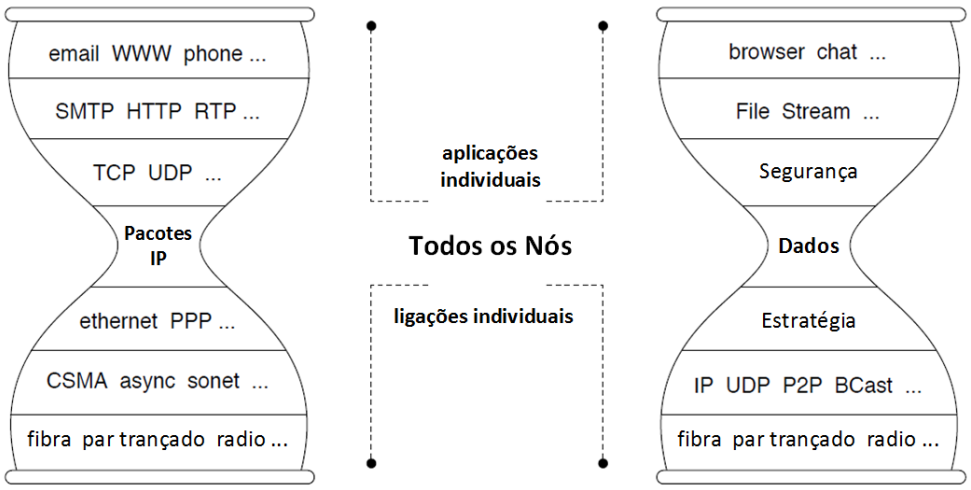


Figura 10 - Diferença das Pilhas Protocolares (adaptado de [5])

A arquitetura das NDNs baseia-se no mesmo principio que a arquitetura da Internet atual. Nas NDNs é também utilizado o princípio da ampulheta, pelo que esta arquitetura pode ser aplicada sem que haja um grande impacto, sendo que a única diferença reside na zona estreita da ampulheta. No protocolo IP são utilizados pacotes que transportam endereços de origem e de destino, já nas NDNs os pacotes transportam nomes de dados sem especificar onde esses mesmos dados se possam encontrar [24]. Através da Figura 10 pode ver-se as semelhanças no princípio de ampulheta utilizado na arquitetura atual (à esquerda) e na arquitetura proposta pela NDN (à direita).

As NDNs baseiam-se num paradigma publica/subscreve, o que faz com que a sua arquitetura se baseie não só numa distinção entre nós produtores e nós consumidores, como também em dois tipos de pacotes: pacotes de dados e pacotes de interesse.

Posto isto, quando um nó consumidor pretende receber algum tipo de informação coloca o nome dos dados desejados num "pacote de interesse" e envia-o para a rede. Por sua vez, os encaminhadores utilizam essa informação (isto é, o nome dos dados) para encaminharem o pacote de interesse até ao nó produtor que produz essa informação. Assim que o pacote de interesse atinge um nó com essa informação, o nó retorna um pacote de dados que contém o nome e conteúdo, juntamente com uma assinatura do produtor, garantindo assim que os dados enviados são verdadeiros. O pacote de dados segue o caminho inverso que o pacote de interesse percorreu, ou seja, se o pacote de interesse foi encaminhado pelos encaminhadores A-B-C, o pacote de dados seguirá o caminho C-B-A, não sendo envolvidos outros encaminhadores.

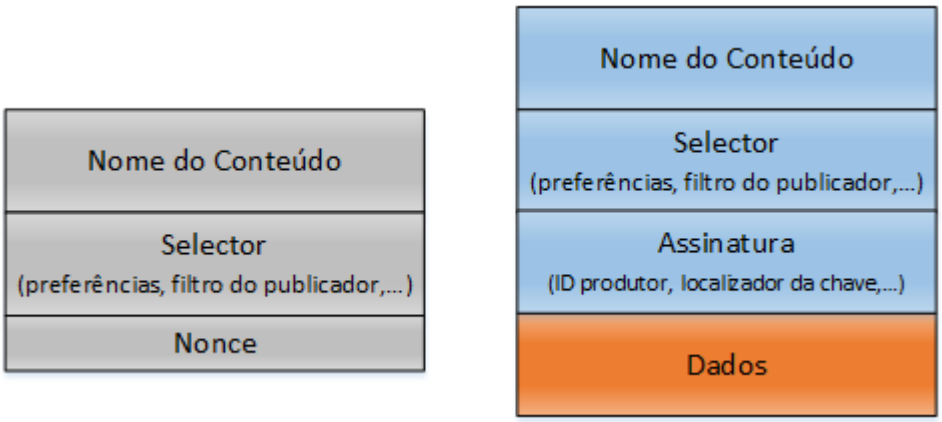


Figura 11 – Pacote de Interesse (à esquerda) e Pacote de Dados (à direita) (adaptado de [25]**)**

Por forma a ser capaz de encaminhar os diferentes pacotes e utilizar esta estratégia de pública/subescreve, cada um dos encaminhadores possui três estruturas distintas: *Pending Interest Table*[10] (PIT), *Forwarding Information Base*[11] (FIB) e a *Content Store*[12] (CS). Na Figura 12 é possível verificar um exemplo prático do estado e da função de cada um destes diferentes componentes.

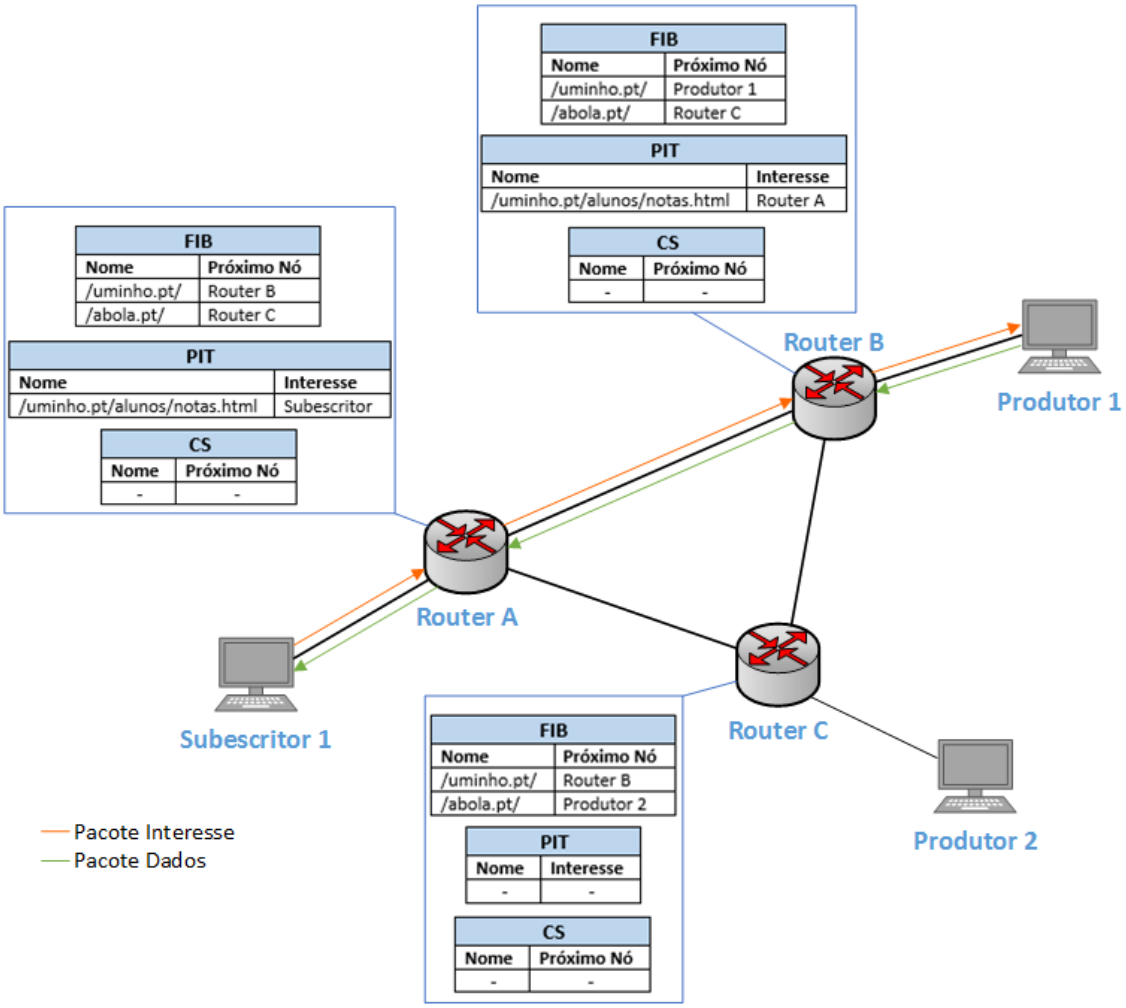


Figura 12 - Exemplo do estado das tabelas nas NDNs [adaptado de [26]]

A PIT não é mais que uma tabela na qual são registados não só os pacotes de interesse que foram encaminhados mas que ainda não foram satisfeitos, como também a interface por onde

[10] Em português: Tabela de Interesses Pendentes
[11] Em português: Base de Informação de Encaminhamento
[12] Em português: Armazém de Conteúdos

esse pacote chega ao encaminhador. Este facto permite que o pacote de dados percorra o mesmo caminho que o pacote de interesse que originou o seu envio, mas no sentido inverso.

A FIB é como que uma base de dados presente em cada encaminhador que indica por qual interface (face) deve seguir cada pacote de interesse. Ou seja, na FIB estão registados os produtores de determinada informação. Assim que um pacote de interesse chega, esta base é consultada por forma a saber qual a interface correta, qual a interface que leva aquele pacote de interesse ao produtor correto.

Por fim, a CS é como que um armazém onde são guardados pacotes de dados temporariamente. Desta forma é possível responder rapidamente a pacotes interesse que cheguem ao encaminhador.

Quando chega um pacote de interesse ao encaminhador o primeiro passo é verificar se existe algum pacote de dados que corresponda àquele interesse armazenado na CS. Caso exista, é imediatamente enviado pela mesma interface pela qual chegou o pacote de interesse. Caso não exista nenhum pacote de dados que satisfaça aquele pedido é então verificada a PIT. Se já existir algum registo de um interesse com aquele nome, apenas é registada a interface pela qual chegou o pacote de interesse. Isto porque, caso já exista um registo, significa que o pacote de interesse com aquele nome já foi encaminhado para o produtor, contudo, ainda não chegou o pacote de dados correspondente. Assim que o pacote de dados chegar, este é replicado e enviado através de ambas as interfaces registadas na PIT. Caso esse interesse não esteja registado na PIT, é então feito esse mesmo registo, bem como qual a interface pela qual chegou esse pacote de interesse. De seguida deve ser consultada a FIB, que indica por qual interface se deve encaminhar determinado interesse. Caso não exista informação na FIB acerca do produtor o pacote de interesse é descartado. Quando um pacote de dados chega ao encaminhador é verificada a PIT, por forma a verificar qual a interface que havia “pedido” aquela informação. Determinada a interface correta, o pacote de dados é enviado em direção ao destino [5].

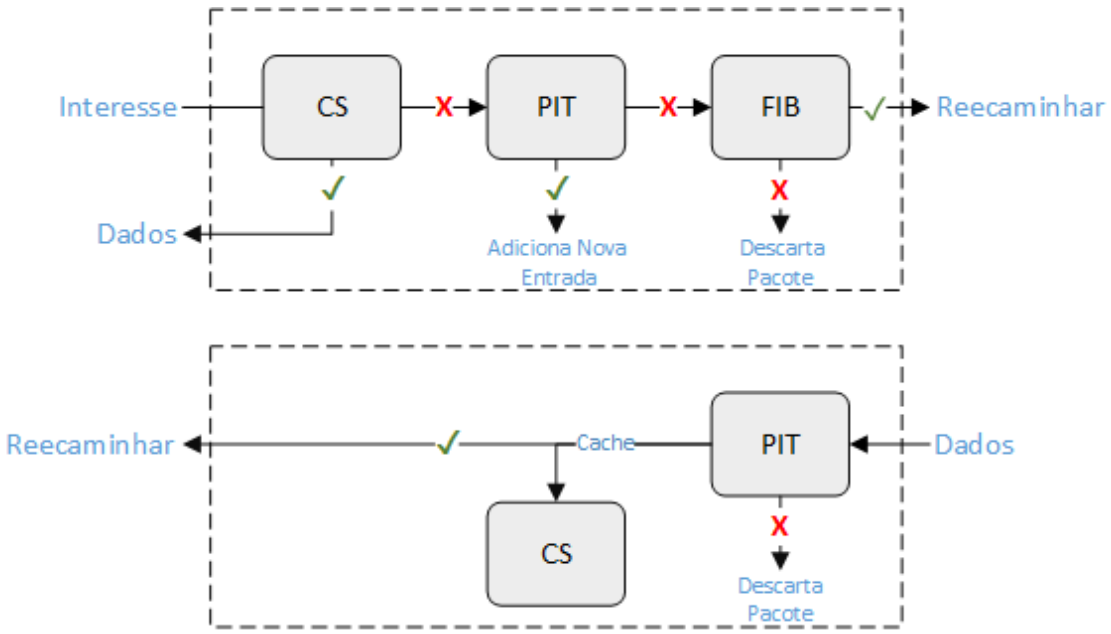


Figura 13 - Processamento de Pacotes Interesse (em cima) e Pacotes Dados (em baixo) (adaptado de[24]**)**

Na implementação original das NDNs, quando um pacote de interesse é enviado para o nó seguinte, é ativado um contador baseado no RTT estimado. Se esse pacote de interesse receber uma resposta antes do contador ter chegado ao fim, o seu valor é atualizado. Porém, existe também a hipótese de um pacote de interesse não receber resposta alguma. Nesse caso, o nó que enviou o pacote de interesse aguarda que o seu contador chegue ao fim, e só depois reporta que foi impossível satisfazer o pedido. Entretanto, o pacote de interesse que não foi satisfeito mantém-se na rede até que o seu tempo de vida termine.

Para "combater" tal problema, foi proposto que se utilize um pacote denominado de "*Interest NACK*". Este pacote, não é mais que uma informação enviada pelo encaminhador que recebeu o pacote de interesse para o encaminhador que o enviou. É então comunicado que o pacote de interesse não pode ser satisfeito. Desta feita, não é necessário esperar que se atinja o fim do contador e o encaminhador que havia enviado o pacote de interesse pode tomar medidas de forma mais rápida. Caso este último encaminhador tenha esgotado todas as suas hipóteses de encaminhar o pacote de interesse, deve ele também enviar um *Interest NACK* para o nó anterior. Um pacote *Interest NACK* contém o mesmo nome que o pacote de interesse que não foi satisfeito, juntamente com um código que indica o motivo de não se poder satisfazer o pedido [27].

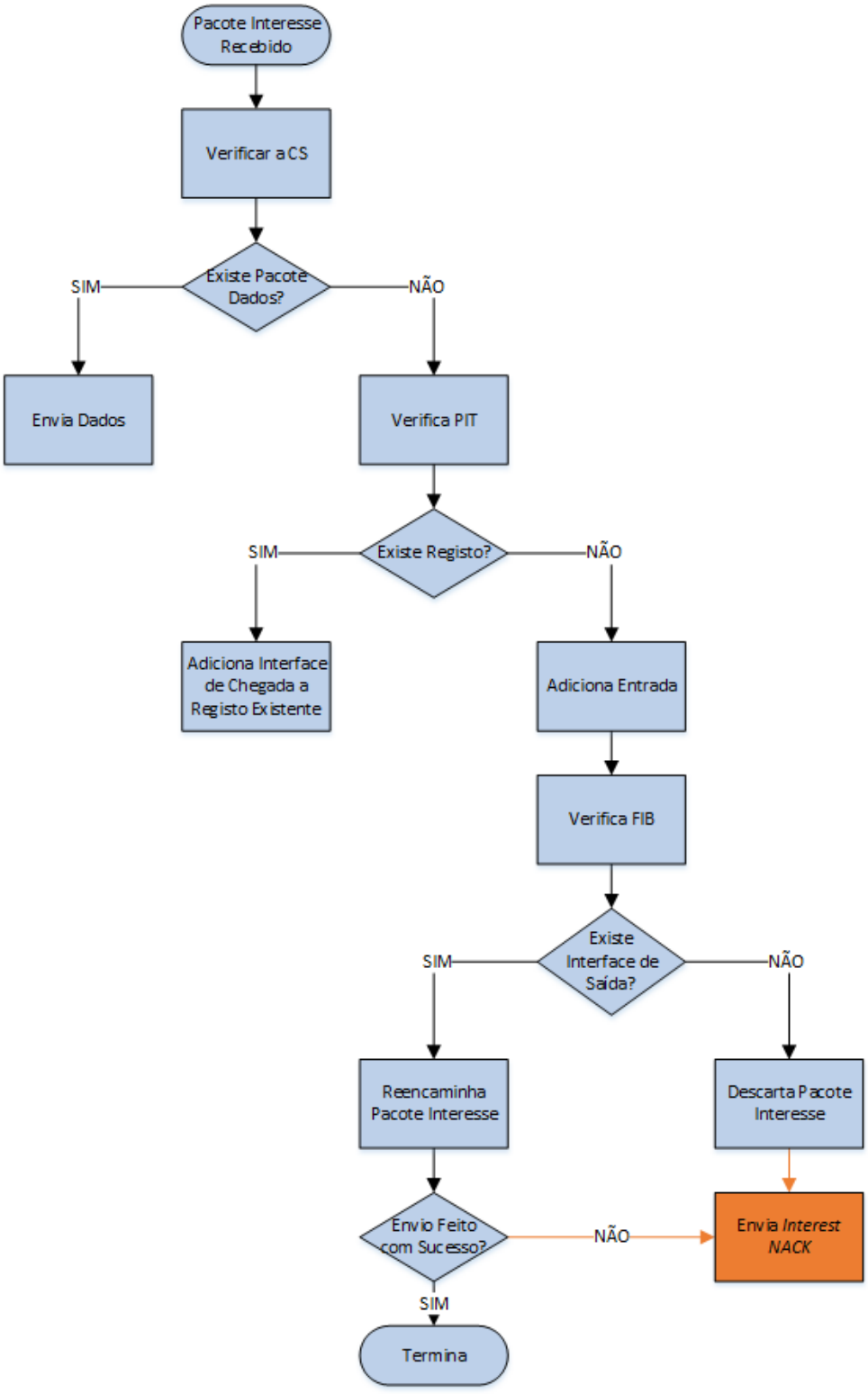


Figura 14 - Processo de Encaminhamento de Pacotes Interesse (adaptado de [5])

No esquemático representado na Figura 14, pode ver-se a cor-de-laranja esta nova proposta da utilização do pacote "Interest NACK". Todo o resto do esquemático representa a abordagem original das NDNs.

3.3. Nomes

O esquema de nomeação dos pacotes é uma das partes mais importantes das NDNs [6]. Para além de identificarem toda a informação que circula na rede, os nomes atribuídos aos pacotes ditam também a facilidade com que um consumidor consegue aceder à informação pretendida.

Apesar de identificarem a informação que circula na rede, os nomes são opacos à rede. Isto é, no seu encaminhamento, os encaminhadores e todos os intervenientes neste processo de transporte dos pacotes não sabem qual o significado do nome atribuído a um determinado pacote. Desta forma, as aplicações têm liberdade para escolher qual o esquema de nomeação com que identificam determinado tipo de informação, independentemente da rede onde estão a operar [6].

Uma das características dos nomes utilizados nas NDNs é o facto de serem hierárquicos. Ou seja, os nomes atribuídos a determinado tipo de informação seguem uma determinada ordem. O primeiro componente de um nome deve ser um sistema autónomo, seguido de uma subdivisão desse mesmo sistema autónomo e por aí adiante. Esses diferentes componentes devem ser separados por o caracter '/', que não faz parte do nome, apenas serve como delimitação entre componentes de um nome [6]. Por exemplo, se um consumidor quiser ver os golos do jogo que opôs Portugal à Itália, o nome deverá ser algo parecido com: abola.pt/videos/PortugalvsItalia/golos.mpg/1/1, no qual "1/1" significa o segmento 1 da versão 1 do vídeo. Esta estrutura hierárquica utilizada no esquema de nomeação permite às aplicações estabelecer relações entre partes de dados, facilitando não só uma futura consulta à mesma fonte, como também a sua gestão. Apesar de terem um aspeto semelhante ao de um URL, o seu primeiro componente não se trata de um servidor DNS ou de um endereço IP e pode nem sequer ser possível a leitura a humanos. Nas NDNs um nome pode ser literalmente qualquer coisa, desde uma palavra que faça sentido até a um valor de *hash* [26].

Os nomes atribuídos aos diferentes pacotes são específicos para cada aplicação que opera nas NDNs. Apesar de não ser obrigatório uma exclusividade a nível global, a atribuição dos nomes devem ter em conta algum grau de exclusividade. Por outras palavras, um nome atribuído a um pacote deve ser tão único quanto maior for a globalidade da comunicação. Para comunicações locais, em redes pequenas, um nome não precisa de ser único quando comparado a um nível global. Deve sim, ser único na sua rede [6]. Os nomes são acertados através de uma convenção prévia entre os produtores e os consumidores. Contudo, por vezes os consumidores têm a necessidade de aceder a informação criada dinamicamente. Como consequência desta

necessidade, não é possível que se estabeleça um acordo entre o produtor e o consumidor no que toca à atribuição do nome para esses mesmos dados. Desta feita, o consumidor deve ser capaz de deterministicamente construir um nome para os dados que deseja receber, sem que tenha antes visto qual o nome que lhes foi atribuído. Para tal, o consumidor pode utilizar uma de duas estratégias, ou até mesmo as duas em conjunto [6]:

1. É utilizado um algoritmo que permite a ambos (produtor e consumidor) chegar ao mesmo nome, partindo da informação que ambos têm disponível. Desta feita, o nome atribuído aos dados por parte do produtor, será o mesmo pelo qual o consumidor os procura;
2. O consumidor tenta aceder aos dados através de parte do nome. Voltando ao exemplo anterior, o consumidor quer ver os golos desse jogo, mas não sabe o nome que deve utilizar. Desta forma, o consumidor pode requerer abola.pt/videos/PortugalvsItalia/golos.mpg e irá receber o conteúdo abola.pt/videos/PortugalvsItalia/golos.mpg/1/1. Através deste dados que foram recebidos, o consumidor é agora capaz de requerer outros segmentos do vídeo.

3.4. Segurança

A segurança nas NDNs é baseada no conteúdo, ou seja, a proteção e confiança são propriedades do próprio conteúdo e não da conexão pela qual os pacotes são transportados. Ao contrário do que acontece nas redes IP utilizadas hoje me dia, nas quais a confiabilidade da informação é baseada na identificação do servidor que as fornece e no tipo de conexão utilizado para o seu transporte [25]. Por outro lado, ao contrário do que acontece no modelo utilizado atualmente para a implementação da Internet, em que as verificações de segurança se fazem aquando o estabelecimento de uma sessão e é feita pelos destinos inicial e final, nas NDNs os mecanismos de segurança estão presentes em cada pacote [5].

Para que a informação contida num determinado pacote seja válida e confiável é necessário verificar três parâmetros: integridade (se o pacote está intacto e completo), pertinência (a que pergunta responde) e proveniência (de onde vem). Para responder a tais questões o modelo de IP utiliza um mecanismo de *checksum*, que de algum modo dá alguma integridade aos pacotes recebidos. Contudo, tanto a pertinência como a proveniência são determinadas baseado no endereço de destino e de origem da informação, respetivamente [25]. Já nas NDNs, a pertinência é determinada através do nome atribuído à informação e a proveniência é assegurada através de uma assinatura do produtor da informação. Em cada um dos pacotes é

acoplada uma assinatura, que é obrigatória. Ou seja, independentemente da aplicação, os pacotes têm de ser assinados por quem gera a informação. Este mecanismo juntamente com o nome pelo qual os dados são identificados e com a informação acerca do publicador conferem uma elevada confiança ao consumidor. Através de todas essas informações presentes em todos os pacotes recebidos, permitem ao consumidor determinar se este é um publicador confiável ou não. Apesar de este mecanismo conferir um elevado grau de confiabilidade, requer uma grande eficiência na geração das assinaturas, transmissão, verificação e no seu possível armazenamento. Contudo, nas NDNs podem ser utilizadas assinaturas utilizando o algoritmo RSA. Por outro lado, nas NDNs não é obrigatório, nem expectável que os encaminhadores verifiquem as assinaturas [6].

Para além deste mecanismo de assinaturas presente intrinsecamente nas NDNs existe também um mecanismo de chave públicas, no qual os consumidores decidem se devem confiar naquele publicador ou não. Para tal, a distribuição da chave pública é feita como se de qualquer outro tipo de informação se tratasse. Ou seja, o produtor envia a chave pública para os consumidores como se enviasse outros dados quaisquer. Contudo, para garantir que a chave pública é válida, ao invés de a sua confiabilidade ser conferida pela assinatura do produtor, é usado um certificado. O facto de os nomes serem estruturados hierarquicamente permite que se aumente a confiabilidade. Recorrendo ao exemplo anterior, abola.pt/videos/PortugalvsItalia/golos.mpg pode ser assinado pelo domínio abola.pt/vídeos, que por sua vez tem a sua chave certificada pelo dominio abola.pt [26].

Posto isto, é possível dizer que as NDNs são baseadas na confiança estabelecida entre o consumidor e o produtor. Contudo, as características da sua arquitetura previnem ainda outros ataques “pessoais”. Isto é, o facto de durante a troca de informações não se utilizar qualquer tipo de endereço final (nem para procurar informação, nem para a recuperar), torna difícil que se marque um determinado *host* como alvo de um ataque. Por outro lado, existe ainda a característica que apenas permite que um pacote seja reencaminhado se existir algo pedido correspondente. Ou seja, apenas são reencaminhados para o consumidor pacotes que tenham sido pedidos por ele, deve assim existir um registo na PIT de cada um dos encaminhadores. Desta forma, previnem-se ataques que tencionem inundar a rede e prejudicar o seu funcionamento, como por exemplo ataques de *Denial of Service* (DoS) [6]. Este mecanismo aliado com o facto de se poderem utilizar diversos caminhos para encaminhar a informação permite aos encaminhadores detetar anomalias [5].

Todos estes mecanismos de segurança presentes nas NDNs permitem às aplicações, aos produtores e consumidores uma maior flexibilidade na hora de escolher qual o mecanismo que mais se adequa ao seu funcionamento.

3.5. Armazenamento

Tal como referido anteriormente, uma das características presentes nas NDNs é a capacidade de guardar, durante um determinado período de tempo, dados nos encaminhadores intervenientes no transporte dos pacotes. Para tal os encaminhadores possuem um componente denominado de *Content Store* (CS), que de uma forma básica corresponde ao *buffer* presente nos encaminhadores utilizados atualmente na implementação do protocolo IP.

Posto isto, assim que um pacote de interesse chega a um encaminhador este verifica o seu CS. Caso exista neste um pacote de dados com um nome coincidente ao pacote de interesse recebido, é enviado de imediato um pacote de dados, contendo a informação pretendida, através da mesma interface por onde havia chegado o pacote de interesse. O facto de os pacotes de dados possuírem um nome que os identifica de uma forma persistente permite que estes sejam reutilizados.

Este mecanismo permite uma grande poupança na largura de banda utilizada. Isto na medida em que um pacote interesse pode não precisar de percorrer todo o caminho até ao produtor. Para tal, basta que um dos encaminhadores tenha armazenado no seu CS o pacote de dados pretendido. Por outro lado, caso um pacote de dados se perca no seu caminho até ao consumidor, não é necessário que este seja retransmitido pelo seu produtor. Assim que o pacote de interesse seja retransmitido, este pode ser satisfeito no nó imediatamente anterior ao pacote de dados se ter perdido. Desta feita, não terá de percorrer novamente toda a rede. O mesmo mecanismo permite também uma recuperação mais rápida quando uma ligação se perde. Assim que a ligação é retomada é realizado o mesmo processo. Por fim, diminui drasticamente o tempo de espera, pois os pedidos podem ter uma resposta de forma muito mais rápida. Pode também ser bastante útil caso se verifique uma situação em que vários consumidores estão interessados nos mesmos dados, num curto espaço de tempo.

Apesar de todas estas vantagens, o facto de se armazenar pacotes de dados nos encaminhadores têm também os seus inconvenientes, nomeadamente no que diz respeito à privacidade. No modelo IP é relativamente fácil verificar qual o endereço de origem e de destino de um pacote. Para tal basta verificar o seu cabeçalho. Para verificar o conteúdo do pacote basta

verificar o conteúdo do *payload*. Todavia, nas NDNs é ainda mais fácil verificar qual o conteúdo de um determinado pacote, bastando verificar o nome que identifica a informação transportada. Já verificar de onde veio e quem o solicitou é um pouco mais difícil que no modelo IP. É possível "aprender" quem recebe o quê sondando o conteúdo presente no CS de um encaminhador. É necessário recorrer a este mecanismo de sondagem pois assim que um pacote de dados é guardado toda a informação relativa a quem o solicitou é apagada [5], [6]. Apesar deste problema, as NDNs apresentam, ainda assim, uma maior privacidade comparando com o modelo atual da Internet.

Este continua a ser um dos alvos de maior estudo nas NDNs, uma vez que existem ainda várias lacunas relacionadas com o armazenamento de pacotes. Por exemplo, é necessário utilizar uma estratégia viável no que diz respeito à gestão dos CS. Não é possível para um encaminhador guardar todos os pacotes que a ele chegam. Sendo assim, é necessário descartar alguns dos pacotes de dados. A grande lacuna reside em quais devem ser mantidos no CS por mais tempo e quais devem ser eliminados assim que o CS encher [26].

3.6. Encaminhamento nas NDN

O próprio funcionamento das NDNs sem recorrer a qualquer tipo de protocolo de encaminhamento proporciona a estas redes uma boa eficiência. Porém, existe ainda espaço para melhorar a sua eficiência e a rapidez com que reagem a alterações na rede. Quer devido a congestionamento em determinados momentos, quer a ligações que se perdem, entre outros. Um outro motivo para a utilização de um protocolo é definir como devem proceder os encaminhadores assim que recebem um pacote de interesse. É necessário que os pacotes de interesses sejam encaminhados pela interface correta, por forma a serem satisfeitos mais rapidamente. Para tal, hoje em dia, vários protocolos de encaminhamento são propostos para as NDNs, sendo que, este tema é ainda alvo de um estudo intensivo.

3.6.1. NLSR: Named-data Link State Routing Protocol

O protocolo de encaminhamento NLSR [28] é um protocolo baseado no estado das ligações. Tal como acontece no OSPF padrão [29], utilizado na arquitetura IP, cada um dos encaminhadores recebe vários LSAs (*Link State Advertisments*[13]). Ou seja, existe como que uma

[13] Em português: Anúncios do Estado das Ligações

inundação da rede com pacotes LSAs por forma a garantir que todos os encaminhadores os recebem. Através da informação contida nesses pacotes, os encaminhadores constroem uma tabela de encaminhamento baseada numa topologia. Desta forma, tendo em conta o estado das ligações conseguem definir qual o próximo salto para cada um dos pacotes recebidos. Este é, de certa forma, o funcionamento de um protocolo baseado no estado das ligações. Quando um encaminhador deteta a queda ou recuperação de uma ligação sua ou de um vizinho seu, este inunda a rede com pacotes LSA contendo atualizações.

Contudo, dado que o NLSR é um protocolo que corre sobre uma rede NDN, para além de permitir aos encaminhadores construir uma topologia (como acima descrito) permite também distribuir prefixos de um nome. Isto é, o encaminhador NDN no qual está implementado o protocolo NLSR, anuncia à rede qual os prefixos dos nomes contidos na sua FIB. As entradas da FIB podem ser introduzidas quer de forma estática através do próprio protocolo NLSR, quer de uma forma dinâmica, através de um qualquer produtor diretamente ligado ao encaminhador. Como tal, sempre que um prefixo de um nome é adicionado ou eliminado da FIB de um encaminhador este dissemina pela rede um novo pacote LSA. A última versão das LSAs é guardada na LSDB (*Link State Database*[14]).

A grande diferença do protocolo NLSR para um protocolo baseado nos estados das ligações, como o OSPF, é o facto de não anunciar apenas o melhor próximo salto. Neste protocolo existem vários próximos saltos, sendo que estes são ordenados. Ou seja, o encaminhador tenta enviar determinado pacote através da melhor ligação, contudo, caso esta não esteja disponível, tenta a segunda melhor e por aí adiante até esgotar todas as possibilidades.

Uma vez que se trata de um protocolo a operar nas NDNs, o registo do melhor próximo salto é feito na FIB. Um encaminhador que recebe um determinado pacote de interesse, depois de verificar a CS e a PIT consulta a FIB e tenta encaminhar o pacote pela melhor interface. Contudo, se esta não estiver disponível, tenta a segunda melhor interface para encaminhar pacotes com aquele nome. No fundo, com este protocolo, podem existir várias interfaces válidas para encaminhar pacotes com o mesmo nome. O que varia entre uma interface e outra é o custo, ou seja, o número de saltos, RTT mais elevado, etc…

Para disseminar as alterações na rede, o protocolo NLSR utiliza pacotes de interesse/dados. Ou seja, utiliza as próprias características da rede para espalhar informação acerca do estado da mesma. Posto isto, os encaminhadores são identificados com nomes e não com IPs como acontece na arquitetura atual. Por outro lado, o facto de utilizar este mecanismo, permite aos

[14] Em português: Base de Dados do Estados das Ligações

encaminhadores confiar nas informações recebidas nos LSAs. Isto porque, tal como referido anteriormente, todos os pacotes de dados a circular na rede são devidamente assinados por quem os produziu. Desta forma, a informação acerca do estado das ligações é de confiança. É garantido assim que um LSA é assinado apenas pelo encaminhador que o gerou, e não foi modificado ao longo do seu percurso.

3.6.2. OSPFN – Protocolo OSPF adaptado às NDNs

O protocolo OSPFN [30] foi o primeiro protocolo a ser implementado nas NDNs [5]. Tal como o protocolo apresentado anteriormente, este é também um protocolo baseado no estado das ligações. Foi desenvolvido pelo mesma equipa que desenvolveu o protocolo NLSR, sendo que este é como que um *upgrade* do OSPFN.

O protocolo OSPF *standard* suporta LSAs opacas (OLSA), por forma a permitir extensibilidade para um uso futuro. Uma OLSA não é mais que um pacote com um cabeçalho de um LSA normal seguido de dados específicos provenientes de uma aplicação. Se existir um encaminhador no qual não esteja a correr a aplicação, este não utiliza esse OLSA para construir a sua topologia. Ainda assim, o OLSA é encaminhado como se de um LSA normal se tratasse. Esta característica do protocolo OSPF é utilizada, na versão para as NDNs, para anunciar o prefixo dos nomes. Através da utilização das OLSAs, é possível anunciar os prefixos dos nomes a toda a rede [30].

Por forma a implementar este protocolo numa NDN cada um dos encaminhadores terá que correr paralelamente três componentes distintos: o CCND (*Content Centric Network Daemon*[15]), o OSPFN e o OSPFD (OSPF *daemon*). O OSPFN é responsável por criar os OLSAs que contêm os prefixos dos nomes. Depois de serem criados, estes OLSAs são injetados no OSPFD que os espalha pela rede. Para além de ter a função de espalhar os OLSAs pela rede, o componente OSPFD é também responsável pela sua receção. Quando tal acontece, este é entregue ao OSPFN. Uma vez que cada LSA contém sempre o ID do encaminhador que o criou, é então possível identificar qual o encaminhador que contém dados com determinado nome. Posto isto, o componente OSPFN questiona o componente OSPFD acerca de qual deverá ser o próximo salto para atingir esse encaminhador. Convém não esquecer que o OSPFD continua a processar os LSAs normais, permitindo criar uma topologia, tal como acontece no OSPF padrão. Sendo assim, e baseando-se na topologia criada, o OSPFD indica o caminho mais curto. Essa informação é

[15] Em português: Daemon da Rede Centrada na Informação

depois guardada no CCND que contém o nome atribuído a determinada informação e qual o próximo nó para o qual deve ser enviado determinado pacote [30].

Desta forma, todos os encaminhadores a operar na NDN podem construir a sua topologia, não só baseada no estado das ligações, mas também no nome dos dados que pretendem receber/enviar.

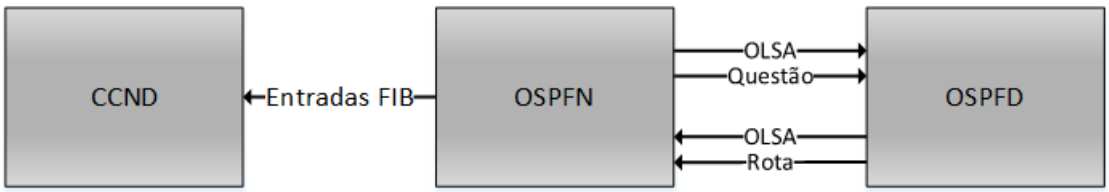


Figura 15 - Interação entre Componentes num *router* OSPFN (adaptado de [30]**)**

Através da Figura 15 é possível compreender mais facilmente as interações que os diferentes componentes têm entre si. Posto isto, verifica-se que o componente OSPFN cria os OLSAs e envia-os para o OSPFD que os mete a circular na rede. Por outro lado, quando é recebido um OLSA, o componente OSPFD recebe-o e envia-o para o OSPFN que identifica qual o encaminhador que o criou. De seguida, o OSPFN questiona o OSPFD qual o melhor caminho para atingir esse encaminhador. É então recebida a melhor rota, ou seja, a rota mais curta, para atingir esse encaminhador. Isto de acordo com a topologia criada pelo OSPFD que lida com os LSAs comuns presentes no protocolo OSPF padrão. Quando obtém a resposta acerca do melhor caminho, o OSPFN guarda essa informação no CCND. Desta feita, fica então guardado no CCND o "nome" dos dados e qual o próximo nó tendo em vista atingir o encaminhador que contêm essa informação.

3.6.3. COBRA

Tal como dito anteriormente, as NDNs baseiam-se em nomes para que os seus utilizadores acedam a um determinado tipo de informação, sendo que esses nomes são nomes hierarquizados. Posto isto, para que um utilizador aceda a um determinado tipo de informação deve então enviar um pacote de interesse contendo o nome (hierárquico) dessa mesma informação na qual ele está interessado. Como consequência, as operações de encaminhamento e de procura de informação são, também elas, baseadas em nomes hierárquicos [31].

Tendo este fator em conta, surgiu um novo protocolo de encaminhamento denominado de COBRA (*COntent-driven Bloom filter based Routing Algorithm*, [31]). Neste protocolo é explorada uma característica das NDNs que até então não havia sido explorada. Tendo em conta que as NDNs têm a capacidade de armazenar dados ao longo da rede (*caching*), os autores defendem um novo conceito. Ou seja, segundo os autores, um protocolo de encaminhamento baseado em nomes deve não só difundir informação acerca de dados que estão guardados de forma permanente, como também deve ser capaz de descobrir cópias temporárias que poderão estar armazenadas noutras rotas, que não as que vão de encontro à origem da informação. AO contrário do que acontece no protocolo NLSR abordado anteriormente. Nesse protocolo as FIBs eram preenchidas apenas com rotas de fossem especificamente de encontro a conteúdos armazenados de forma permanente. Os dados armazenados temporariamente ao longo da rede não são tidos em conta [31].

O protocolo de encaminhamento COBRA é um protocolo intradomínio baseado na estrutura de dados probabilística Bloom Filter [32]. Neste protocolo é explorado um conceito que consiste em deixar como que um rasto dos pacotes de dados que vão em direção aos nós que os solicitaram. Para tal, assim que um pacote de dados é recebido é feito um *hash* do seu prefixo juntamente com a interface pela qual foi recebido. Esta abordagem torna este protocolo: simples, uma vez que não necessita qualquer outra estrutura de dados ou processamento de pacotes que poderia aumentar a carga computacional dos cabeçalhos numa rede NDN; eficiente, uma vez que não necessita de qualquer tipo de mensagem de atualização; autónomo, dado que não necessita de nenhum tipo de encaminhamento IP a operar como mecanismo de prevenção ou a anunciar rotas para cópias permanentes [31].

Por forma a processar os pacotes recebidos e deixar como que um rasto de por onde eles passaram, o protocolo instala um *Bloom Filter* em cada interface de cada um dos nós NDN. Sempre que um pacote de dados é recebido é encaminhado para o cliente que o solicitou, como seria de esperar. Após esta fase, é então feito o cálculo de um *hash* que conjuga a totalidade do nome do pacote encaminhado com a interface pela qual havia chegado. Esse resultado é guardado. De seguida é feito o mesmo cálculo de um *hash* mas desta feita conjugando apenas parte do nome com a interface pela qual o pacote de dados chegou. Ou seja, é retirado o último componente do nome. Por exemplo, se um pacote de dados tem o nome /uminho.pt/di/dissertações é feito o cálculo da *hash* para o nome completo, de seguida é feito o mesmo cálculo para parte do nome (/uminho.pt/di). Este processo repete-se até que não existam mais prefixos do nome disponíveis. Desta feita, o caminho criado pode ser utilizado para

encaminhar pacotes de interesse que partilhem a totalidade ou parte do nome do pacote de dados anteriormente encaminhados [31].

Capítulo 4 – Convergência entre DTNs e NDNs

4.1. Introdução

Após uma descrição geral de cada uma das redes, das suas arquiteturas e do seu funcionamento é agora chegado o momento de analisar os pontos nos quais estas convergem. Ou seja, apesar de utilizarem abordagens diferentes e das suas arquiteturas apresentarem diferenças acentuadas, existem também pontos em comum entre estas duas redes.

Através da análise feita anteriormente pode perceber-se que as DTNs apresentam um paradigma bastante flexível e resiliente em ambientes adversos, nos quais não são garantidas ligações entre os diferentes componentes da rede, não só devido à elevada mobilidade dos nós, mas também à possível ausência de infraestruturas. Como resultado destas adversidades surgem atrasos na transmissão dos dados, taxas de erro elevadas ou, em casos extremos, a impossibilidade de comunicação. É utilizado um paradigma de *store-carry-and-forward* para que as mensagens possam seguir em direção ao seu destino final. Apesar de não ser tão claro como nas redes IP, as DTNs são ainda assim focadas em entidades, pelo que se torna necessária a identificação de cada uma delas. O encaminhamento é feito tendo em conta os nós, encaminhando *bundles* de nó em nó sem que seja tido em conta o conteúdo de cada um deles [4].

Por outro lado, as NDNs são redes direcionadas para a distribuição de conteúdos. Utilizando uma abordagem completamente diferente da rede IP convencional, é centrada na informação que circula na rede e não nos nós ou entidades que a compõem. É utilizado o paradigma publica/subscreve, em detrimento da abstração do conceito “fim-a-fim”. O facto de se focar na informação permite que esta seja armazenada e posteriormente reutilizada, caso seja necessário. Desta forma, existe também mais apoio a uma possível mobilidade dos nós.

Tal como referido anteriormente, apesar destas diferenças, existem também pontos comuns a ambas as redes. Nomeadamente, ambas as redes permitem armazenamento de dados na rede, ainda que com propósitos diferentes. Nas DTNs o armazenamento serve como um meio para um posterior envio, caso não seja possível enviar em determinado momento. Nas NDNs o

armazenamento é visto como um meio de reduzir a latência, uma vez que permite que os conteúdos sejam reutilizados. Ambas as redes permitem a utilização de rotas diferentes para chegar ao mesmo destino e ambas possuem mecanismos de deteção de *loops*.

Através da análise da Tabela 1 pode ver-se que existem vários pontos comuns às duas redes. Partindo destes pontos comuns, uma fusão entre elas é logicamente o caminho que pode ser seguido. Desta forma, a rede resultante, teria um potencial enorme, uma vez que juntaria os benefícios de ambas. Eventualmente, esta junção apresentaria uma melhor resposta à mobilidade dos nós e aos atrasos provocados pela mesma, quando comparada com as NDNs, e seria também mais rápida no que diz respeito à entrega de conteúdos, em relação às DTNs.

Tabela 1 – Comparação entre DTNs e NDNs (adaptado de [4])

Característica	Delay Tolerant Network	Named Data Network
Relação com a Internet	Concebida para funcionar me cenários em que a Internet não funciona. Pode funcionar como rede de sobreposição.	Concebida para substituir a Internet. Pode coexistir com a Internet, funcionando como *overlay* ou como sub-rede da Internet
Interrupções	Utiliza o conceito "*Story-Carry-and-Forward*". As mensagens são armazenadas e encaminhadas posteriormente.	Pacotes de dados são descartados se não existir entrada na PIT. Pacotes interesse são descartados se não existir entrada na FIB.
Nomes e Endereços	Utilizado esquema URI no endereçamento dos nós.	Nomes atribuídos a conteúdos de forma hierárquica. Não usa endereços.
Armazenamento	É tido como parte da ato de entrega (*story-carry-and-forward*); Temporário.	Temporário quando em *cache*; Mais persistente na Repo.
Fragmentação	Possível em todos os nós, aquando uma retransmissão.	Apenas possível na origem.
Modelo	Push	Pull
Segurança	Opcional, quer a nível de canal de transmissão, quer a nível de conteúdo.	Apenas existe a nível de conteúdo, sendo obrigatório. Todos os conteúdos são assinados pelo respetivo produtor.
Multicast	Permite *multicast*	Permite *multicast* apenas na entrega de dados

Controlo de Fluxo e Confiabilidade	Utiliza o conceito de custódia.	Apenas destino final poderá comunicar que determinado interesse não foi satisfeito.
Desempenho de Entrega	Não foi concebido com este propósito.	Entrega de dados rápida por forma a reduzir atrasos.

4.2. Trabalhos Realizados

Uma vez que esta solução, de fundir as duas arquiteturas, tem sido um caso de estudo bastante popular, surgiram ao longo dos últimos anos alguns trabalhos que relacionam as duas redes. São avaliadas as vantagens e desvantagens de cada uma das redes com o objetivo de chegar à melhor combinação possível.

4.2.1. Social-Tie based Content Retrieval – STCR

Com o aparecimento das *Sparse MANETs*[16] surgiu a necessidade de distribuir os dados por entre os diferentes nós que compõem a rede. Uma *Sparse MANET* é uma subclasse das redes MANET comuns. Contudo, neste tipo de MANET os nós estão bastante dispersos e as ligações que existem entre si são bastante limitadas, podendo não durar muito tempo. No fundo, este tipo de rede é um exemplo de uma rede DTN, em que as ligações entre os nós da rede são bastante dinâmicas e variam rapidamente. Neste cenário, é muito pouco provável que em algum momento a rede se encontre toda ela ligada entre si. Um grande desafio neste tipo de MANET é a habilidade que a rede deverá ter para distribuir grandes quantidades de dados. Como dito anteriormente, as redes centradas na informação têm facilidade na distribuição de grandes quantidades de dados.

Posto isto, surge assim o algoritmo STCR (*Social-Tie based Content Retrieval*, [33], [34]). Este algoritmo utiliza o mesmo conceito de publica/subescreve presente nas redes centradas na informação. Contudo, devido ao facto de os nós estarem bastante dispersos é utilizado também o conceito de *carry-store-and-forward* presente nas DTNs.

Para evitar a inundação da rede com pacotes desnecessários para grande parte dos nós, é utilizado princípio de encaminhamento social. Este princípio é utilizado em alguns casos nas DTNs. Neste princípio é atribuído um determinado nível de popularidade a cada um dos nós, dependendo da quantidade de nós com que contacta. Ou seja, se um nó "A" se cruzar com vários

[16] Em portugês: Redes Moveis ad-hoc Dispersas

nós ao longo do tempo, então, tem um elevado nível de popularidade. Caso esse nó "A" se cruze com poucos nós, então "A" popularidade baixa. Para determinar um nível de popularidade, sempre que dois nós se encontram, trocam entre si um vetor no qual indicam com quem se cruzaram. Quantos mais encontros um nó realizar, mais popular será. Parte-se do princípio que se dois nós se cruzam a probabilidade de se voltarem a encontrar futuramente aumenta. Os pacotes a circular na rede são assim direcionados para os nós mais populares, uma vez que estes têm mais probabilidade de encontrar o nó de destino. Cada um dos nós tem a sua própria tabela de popularidade e constrói uma tabela em que os nós mais populares se encontram no centro [34].

Esta maior centralidade por parte de um determinado nó, apenas significa que este nó tem uma maior probabilidade média de se encontrar com todos os outros nós. Se for conhecido o destino de uma mensagem, o facto de a encaminhar para os nós mais centrais pode não significar uma maior taxa de entrega no que diz respeito àquele nó específico (destino da mensagem). Isto porque, uma maior probabilidade média de encontrar outros nós, pode não se verificar para aquele nó em específico. É necessário verificar quais os nós mais úteis e quais não o são apesar de estarem num nível de maior popularidade. Essa utilidade varia de entrega para entrega [34].

Por forma a contrariar esses aspetos, o algoritmo STCR não se baseia apenas na frequência dos contactos, mas também na sua frescura. São então utilizados dois critérios para definir a popularidade de um determinado nó: por um lado a frequência dos contactos e por outro a sua frescura. A frescura diminui quando aumenta o tempo desde o último contacto.

Por outro lado, o STCR apenas utiliza o conceito de centralidade para descobrir quem é que possui determinado tipo de informação. Depois desta etapa é utilizado um processo de relações sociais. Ou seja, o pacote de interesse é encaminhado para os nós que tenham mais relações sociais com o nó produtor. Por fim, baseando-se no algoritmo K-mean [35], os diferentes nós são divididos em grupos (*clusters*). Desta forma é mais fácil identificar qual a direção correta tendo em vista um melhor desempenho no que diz respeito aos custos de transmissão.

Sempre que dois nós se encontram é criado um vetor que indica o ID do nó com o qual ocorreu o encontro e o momento em que tal aconteceu. Desta forma, é possível determinar a frequência com que dois nós se encontram e o quão recentemente eles se encontraram. Cada um dos nós possui assim uma estrutura, na qual estão guardados um conjunto de vetores, sendo que, cada um dos vetores corresponde a um encontro. Desta feita, cada um dos nós, tendo em

conta a frequência com que se encontra com um outro nó e os intervalos de tempo entre encontros, define um nível social para caracterizar a sua relação.

Para além de proceder ao registo acima referido e ao cálculo do nível de relação social, cada um dos nós deve também formar uma tabela. Nessa tabela fica registado o nível de relação social que determinado nó tem com cada um dos nós que encontrou. Esta tabela deve ficar ordenada de forma decrescente. Ou seja, se um nó "A" tem um elevado nível de relacionamento com um nó "B", então, "B" deve ficar no topo da tabela de relacionamento de "A".

A tabela deve ser atualizada a cada encontro realizado com qualquer nó. Quando dois nós se encontram trocam essa tabela. Desta forma, ambos podem atualizar as suas tabelas, alargando assim o seu conhecimento acerca da rede. Por fim, tendo como base a sua tabela de relacionamentos, cada um dos nós pode calcular a centralidade dos nós. Isto é, pode calcular quais os nós mais populares.

Convém relembrar que os nós com maior centralidade têm maior probabilidade de encontrar outros nós.

Resumindo, cada um dos nós: calcula o nível de relação social que tem com outros nós que encontra; organizar uma tabela em que figuram todos os outros nós por ordem decrescente de relação social; troca e atualiza a sua tabela à medida que encontra outros nós; e, por fim, calcula a centralidade dos nós tendo em conta a tabela que possui.

Para evitar que a rede seja inundada de pacotes interesse provenientes dos consumidores, os nós enviam um resumo dos conteúdos que possuem. Esse resumo é enviado em direção aos nós mais centrais, uma vez que estes têm mais probabilidade de encontrar outros nós. Ou seja, se um nó possui dados, envia um resumo dos dados que tem aos nós com um nível de centralidade superior à sua. Nesse resumo são explícitos quais os conteúdos que determinado nó possui, o seu ID e o tempo em que esse resumo foi feito.

Por este facto, cada um dos nós presentes na rede possui uma tabela na qual são guardados resumos provenientes dos nós de centralidade menor. Quanto maior for a centralidade de um nó, maior será o seu conhecimento acerca dos conteúdos disponíveis na rede. Resumidamente, se um nó encontra um outro nó mais central que ele próprio envia o resumo dos seus conteúdos e a sua tabela de resumos. Se encontrar um nó com uma centralidade mais baixa, limita-se a enviar informação a respeito das relações sociais com outros nós.

Utilizando o algoritmo K-mean [35], os nós são divididos em grupos de acordo com a sua centralidade. Nós com níveis idênticos de centralidade são colocados no mesmo grupo. Desta forma, se dois nós se encontram e um deles tem uma centralidade ligeiramente maior que o

outro (caso estejam no mesmo grupo), não há grande benefício em transferir a informação. Uma vez que a centralidade não varia muito o nível de conhecimento acerca da localização dos conteúdos é praticamente o mesmo. Já se dois nós estão em grupos diferentes, significa que o nível de conhecimento da rede é substancialmente diferente. Nesse caso, faz sentido efetuar-se a transferência de informação.

Posto isto, se um nó tem interesse em determinado conteúdo envia um pacote de interesse, contendo o nome dos dados nos quais está interessado e o seu ID. Este pacote é direcionado para os nós mais centrais, ou seja, nós pertencentes a um grupo superior. Quando um nó recebe um pacote de interesse, verifica a sua tabela de resumos. Se esse interesse constar na sua tabela, então é adicionado ao pacote de interesse o ID do nó que possui esse conteúdo e a busca entra numa nova fase. Caso não haja correspondência, então o pacote de interesse é direcionado para um nó de um grupo superior. Eventualmente irá chegar ao nó mais central de toda a rede e será descoberto o nó que possui a informação pretendida.

Depois de se descobrir qual o nó que possui essa informação a busca é feita tendo em conta as relações sociais com aquele nó. Ou seja, o pacote de interesse apenas enviado para nós que tenham um maior nível de relacionamento social com o nó que possui a informação. Assim que o pacote de interesse chega ao seu destino (isto é, ao nó que possui aquele conteúdo) é enviado para o consumidor o conteúdo que ele pretende. Para tal é utilizada a mesma abordagem. O pacote de dados é encaminhado apenas se determinado nó tiver um maior nível de relacionamento social com o consumidor [34].

Quando um pacote de interesse é criado, é-lhe atribuído um tempo de vida. Assim que esse tempo de vida expira, o pacote é descartado. Por outro lado, quando um nó injeta um pacote de interesse na rede é também iniciado um contador. Se este contador chegar ao fim, são adotadas novas estratégias de encaminhamento. Assim, existem três estratégias para transmitir os pacotes interesse.

Na primeira estratégia o pacote só é encaminhado para um outro nó, caso esse nó pertença a um grupo superior. O pacote só volta a ser transmitido a um nó que seja de um grupo superior do nó para o qual foi transmitido anteriormente. Ou seja, um nó “A” possui um pacote de interesse e encontra-se com um nó “B” que pertence a um grupo superior. “A” encaminha o pacote de interesse para “B”. No instante seguinte, “A” cruza-se com um nó “C”. “A” só transmite o pacote de interesse a “C” se este pertencer a um grupo superior a “B”. Desta forma evita-se a inundação da rede com pacotes interesse.

Caso os dados demorem muito tempo a chegar ao consumidor e o contador chegue ao fim, é enviado outro pacote de interesse, contudo é utilizada uma outra estratégia. Desta feita, um nó transmite um pacote de interesse se o outro nó pertencente a um grupo superior. Não são tidos em conta encontros anteriores. Ou seja, se um nó "A" encontra um nó "B" de um grupo superior, então o pacote de interesse é transmitido para "B". Se "A" encontrar um nó "C" pertencente ao mesmo grupo de "B", o pacote de interesse é igualmente transmitido.

Ainda assim, se o contador do consumidor chegar ao fim, é reenviado um outro pacote de interesse. Desta vez, é utilizada uma estratégia epidémica. Ou seja, o pacote de interesse é transmitido a todos os nós que ainda não possuam uma cópia.

No pacote de interesse estão designados: o ID do consumidor, o nome dos dados nos quais o consumidor está interessado, o seu tempo de vida, o método de encaminhamento que está a ser utilizado, e ainda o ID do produtor (este apenas é preenchido assim que se descobre quem é o nó que possui aqueles determinados dados).

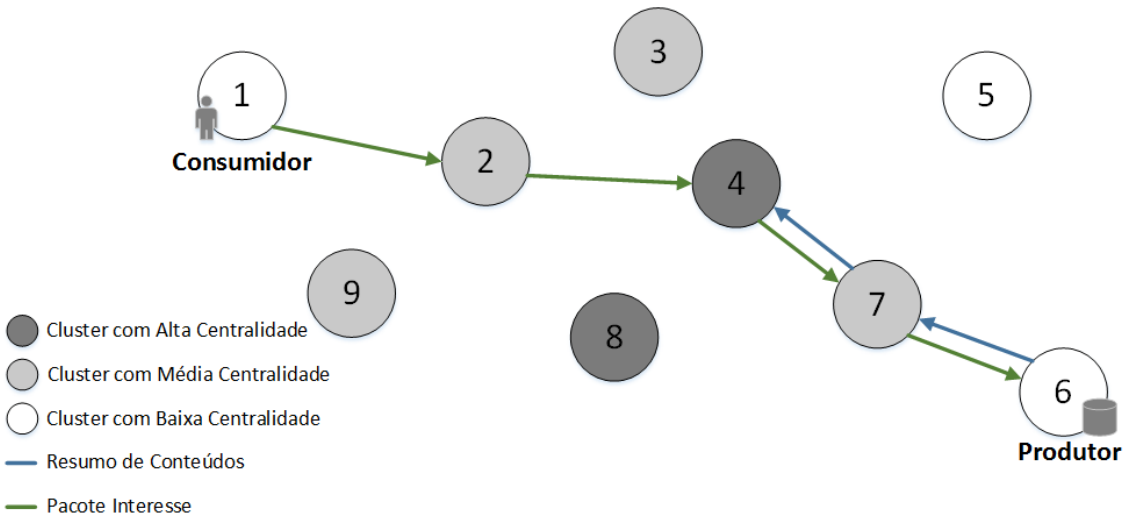


Figura 16 - Fase 1 e Fase 2 do Algoritmo STCR

Através da Figura 16 pode ver-se o comportamento do algoritmo STCR num ambiente real. Nesta figura estão exemplificadas as duas primeiras fases do STCR. A primeira fase, representada pela cor azul, consiste na distribuição dos resumos. O encaminhador 6, após efetuar os cálculos para determinar a centralidade dos nós que encontrou, conclui que o encaminhador 7 tem um nível de centralidade maior que o seu. Desta feita, envia-lhe um resumo dos seus conteúdos. Por sua vez, ao receber esse resumo, o encaminhador 7 atualiza a sua tabela de resumos e

reencaminha o resumo (do seu conteúdo, caso possua algum, e do encaminhador 6) para o encaminhador 4. Isto porque o encaminhador 4 tem um maior nível de centralidade.

A segunda fase, representada a verde, consiste no encaminhamento do pacote interesse. O consumidor envia o pacote para o encaminhador 2, uma vez que este pertence a um grupo de maior centralidade. Por sua vez, o encaminhador 2 verifica a sua tabela de resumos. Uma vez que não existe nenhuma correspondência encaminha o pacote para um outro nó pertencente a um grupo superior, nomeadamente o encaminhador 4. Desta feita, o encaminhador 4 verifica a sua tabela de resumos e encontra uma correspondência. É agora conhecido o nó que contém os dados pretendidos, o encaminhador 6. Então, o encaminhador 4 encaminha o pacote de interesse para o encaminhador 7, uma vez que este tem um nível de relação social com o encaminhador 6 maior que ele próprio. Por fim, o encaminhador 7 encaminha o pacote de interesse para o seu destino final, o encaminhador 6.

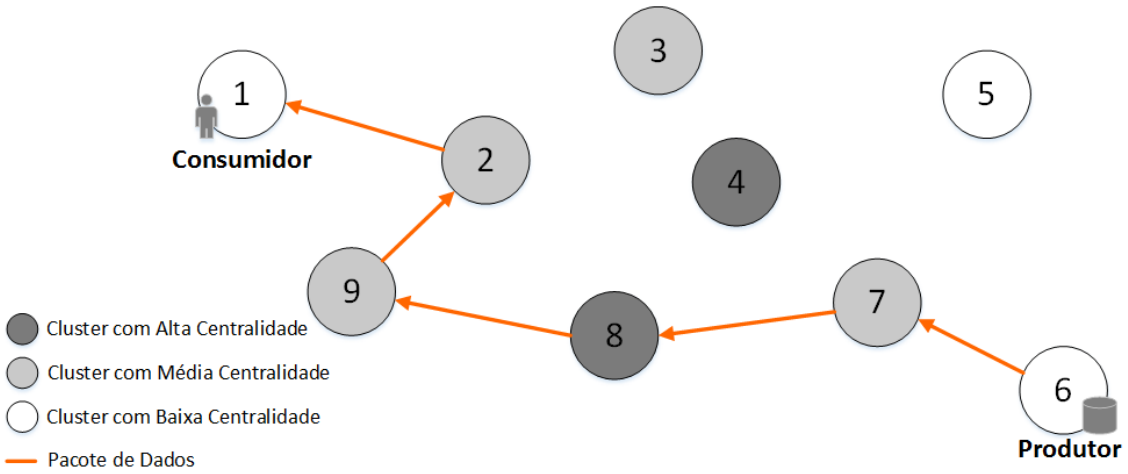


Figura 17 - Fase 3 do Algoritmo STCR

Na Figura 17 está representada a terceira e última fase do algoritmo STRC. Nesta fase o pacote de dados e transmitido desde o produtor até ao consumidor. Para tal apenas é tido em conta o nível de relação social de cada um dos nós com o nó de destino, o encaminhador 1. Desta feita, o produtor encaminha o pacote de dados para o encaminhador 7, por este apresentar, de todos os nós aos quais está ligado, o maior nível de relacionamento com o encaminhador 1. Assim sucessivamente até atingir o seu destino.

4.2.2. Broadcast-Only Named Data – BOND

Ao contrário do que acontece nas redes IP, no protocolo BOND [36] não é definido um caminho prévio para determinado destino. Isto é, antes de se proceder ao envio dos dados não é estabelecido nenhum caminho nó-a-nó até ao destino final. Tal não seria viável devido ao facto de numa rede móvel os nós se movimentarem livremente quer em termos de direção, quer em termos de velocidade. Deste modo, as ligações perdem-se com bastante frequência, podendo apenas estar disponíveis por breves momentos. É então praticamente impossível uma rede móvel completamente conectada.

O BOND [36] é um protocolo que permite a transmissão de pacotes de dados nomeados em redes móveis sem fios. Ou seja, é aplicado às redes móveis esparsas (idênticas às que se podem usar as DTNs) o conceito fundamental das NDNs: a comunicação deve estar centrada nos dados e não nos nós. Isto significa, tal como visto anteriormente, que os dados contidos num pacote não dependem de nenhum nó envolvido numa transmissão em particular. Qualquer um dos nós presentes na rede pode armazenar pacotes de dados que a ele cheguem. Deste modo, poderá satisfazer mais rapidamente futuros pedidos procurando pelos mesmos dados.

Contudo, nas NDNs é também definido qual o melhor nó para encaminhar determinado tipo de informação. Ou seja, é expectável que os nós sejam fixos, não se movimentando. Para poder adaptar a filosofia das NDNs às DTNs este protocolo apresenta vários mecanismos que permitem a comunicação independentemente da topologia e da localização geográfica dos nós.

Tal como acontece nas NDNs, assim que determinado nó requer algum tipo de dados é criado um pacote de interesse, contendo o nome desses mesmos dados. Por cada pacote de interesse recebido, será também enviado, como resposta, um pacote de dados por parte do produtor. Contudo, nas NDNs à medida que um pacote de interesse vai sendo encaminhado ao longo da rede até chegar ao produtor vai deixando como que um rasto. Desta forma, o pacote de dados poderá ser encaminhado no sentido inverso seguindo o rasto deixado pelo pacote de interesse.

No protocolo BOND isso não é possível, porque os nós não se encontram fixos numa determinada posição. Se um pacote de interesse é encaminhado por determinado nó, não é garantido que esse mesmo nó se mantenha na mesma posição. É bastante improvável que o nó receba o pacote de dados no sentido inverso. Para além disso, uma vez que as redes móveis são bastante dinâmicas não é garantida uma ligação desde o produtor até ao consumidor.

Para resolver esta adversidade o protocolo recorre a dois esquemas de nomeação. Tal como acontece nas NDNs, existe um esquema de nomeação para os dados. Porém, existe no protocolo

BOND um outro esquema de nomeação mas desta feita para os nós. Ou seja, os nós são também identificados. O nome de um nó é um identificador único que é utilizado para entregar dados a um determinado nó que os tenha requisitado. Para implementar este novo esquema de nomeação, é utilizado o endereço MAC presente em todos os dispositivos. Os pacotes de dados podem ser encaminhados em direção a qualquer nome [36].

Tal como referido anteriormente, assim que determinado nó pretende receber dados é criado um pacote de interesse. Uma vez que o consumidor não sabe quem possui os dados pretendidos, começa por inundar a rede com cópias do pacote de interesse. Ou seja, o consumidor envia uma cópia do seu pacote de interesse para todos os nós que se encontram ligados a si. Assim que o pacote de interesse atinge um nó que possui os dados em questão, é enviado um pacote contendo esses mesmos dados. Para encaminhar este pacote é utilizada a informação aprendida aquando a transmissão do pacote de interesse, evitando assim que se proceda mais uma vez à inundação da rede. Sendo assim, todos os nós que se encontram no raio de alcance, aprendem não só a localização de determinada informação, como também guardam uma cópia daquele pacote de dados, podendo assim satisfazer futuros pedidos [36].

As decisões de encaminhamento são tomadas assim que um nó recebe qualquer tipo de pacote. Ao receber um pacote, quer seja interesse ou dados, um nó decide para quem deve encaminhar aquele pacote, ou se deve ele mesmo transportar o pacote e posteriormente tentar encaminhá-lo.

Ao receber um pacote, primeiramente, o nó deve decidir se é encaminhador elegível. Ou seja, deve decidir se ele próprio é capaz de ajudar a encaminhar o pacote até ao seu destino final. Caso decida que sim, então esse nó deve aguardar por um determinado período de tempo antes de transmitir o pacote para o nó seguinte. O facto de se aguardar um determinado período de tempo evita possíveis colisões, uma vez que não é muito provável que dois nós aguardem exatamente o mesmo tempo antes de transmitirem o pacote. Esse tempo é escolhido aleatoriamente tendo em conta a distância ao produtor/consumidor. Nós que se encontram mais perto do destino aguardam menos tempo.

Esse tempo de espera é denominado de “período de escuta” e varia de nó para nó consoante a sua proximidade com o destino. Os nós que possuem determinado pacote e se encontram mais próximos do destino têm um período de escuta mais curto. De forma a determinar a distância a que cada nó se encontra da origem de cada nome é utilizada uma determinada métrica. Assim, quando um nó recebe um pacote proveniente diretamente da fonte de um determinado nome, este indica a distância a que se encontra. Essa distância,

dependendo da métrica utilizada, é guardada no cabeçalho do pacote. Mais propriamente no campo, "*srcDist*". Assim que o pacote é encaminhado, o recetor adiciona a distância a que ele se encontra em relação ao nó que o transmitiu. Desta forma, os nós sabem a que distância se encontram de determinado nome. Quando se pretende fazer o sentido inverso, cada um dos nós já sabe a que distância se encontra do destino daquele determinado pacote. Sendo assim, cada um dos nós guarda no campo "*dstDist*", também presente no cabeçalho de cada pacote, a distância a que se encontra do destino [36]. Esta distância permite aos nós determinarem se são ou não elegíveis e calcular a distância a que se encontram do destino.

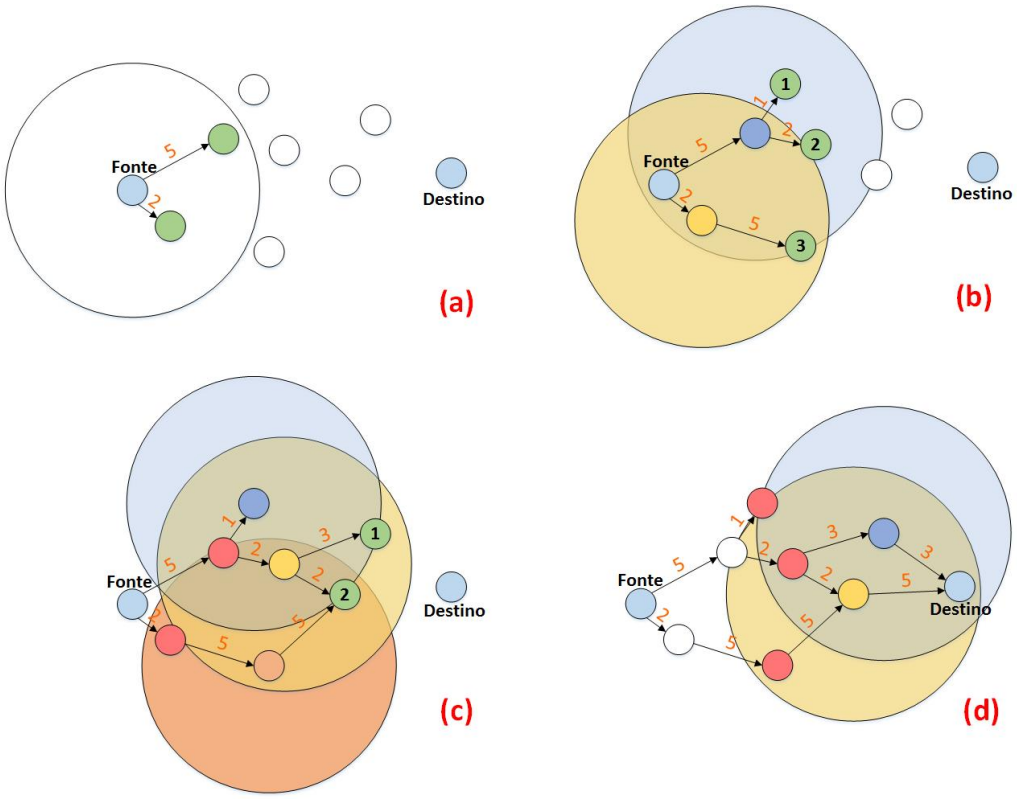


Figura 18 - Funcionamento do protocolo BOND

Através da Figura 18 pode ver-se o funcionamento do protocolo BOND num ambiente mais real. Em (a), um nó denominado de "fonte" (consumidor) pretende receber determinados dados. Para tal, envia um pacote de interesse para todos os nós que estão ao seu alcance. É então estabelecida uma métrica para determinar a distância de cada um dos nós ao nó fonte. A distância é indicada na figura através dos números a cor-de-laranja acima das setas. O processo repete-se em (b) e (c), sendo que os nós a vermelho são nós não elegíveis, ou seja, encontram-se mais longe do destino do que o próprio nó que contem o pacote. Finalmente em (d) o pacote

de interesse chega a um nó capaz de satisfazer esse pedido. Sendo assim, pode verificar-se que dois pacotes de interesse chegaram ao produtor. O produtor apenas responde ao primeiro que recebeu, sendo que deverá optar por um dos nós mais próximos do nó fonte. Em (d), no campo *srcDist* do pacote entregue via o nó azul deverá constar 13. Já no pacote de interesse entregue via o nó amarelo deverá constar 14.

Posto isto, é relativamente fácil utilizar a informação aprendida anteriormente para encaminhar o pacote de dados no sentido inverso.

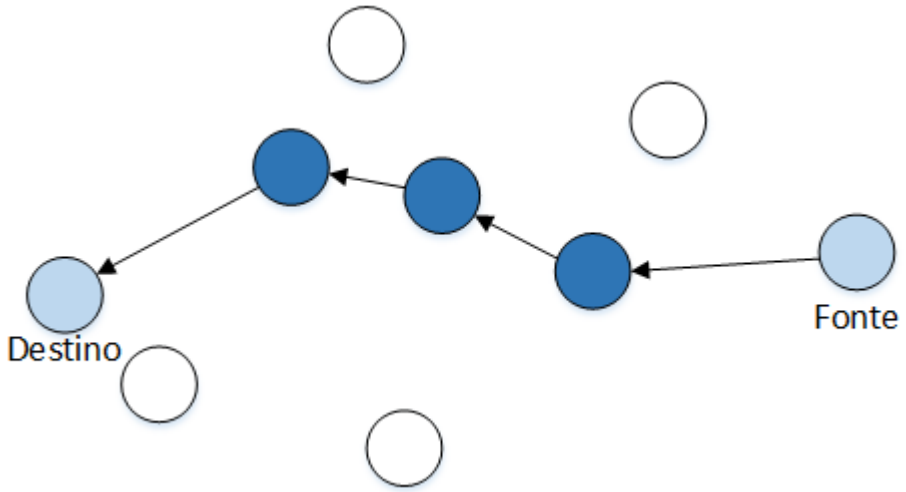


Figura 19 - Funcionamento protocolo BOND (2)

Pode ver-se na Figura 19 qual o melhor caminho tendo em conta a distância ao destino. Uma vez que já eram conhecidas a distâncias de cada um dos nós para um nó com determinado nome, é relativamente fácil verificar qual o melhor nó a quem encaminhar o pacote de dados.

Por fim, existe também um código que identifica cada um dos pacotes transmitidos. Assim, quando um nó recebe um pacote, facilmente determina se se trata de um novo pacote, de um pacote que já reencaminhou ou ainda, se se trata de um pacote ao qual ele próprio já respondeu. Por exemplo, se um nó está no período de escuta e um outro nó encaminha o pacote antes dele, o envio do pacote é cancelado. Esse identificador é também incluído no cabeçalho de cada um dos pacotes.

4.2.3. Listen First Broadcast Later – LFBL

O protocolo LFBL [37] foi desenvolvido para operar em redes sem fios. Este protocolo não necessita de qualquer tipo de rota predefinida para que os pacotes sejam encaminhados desde a sua fonte até ao seu destino. Ao utilizar uma abordagem em tudo parecida com a abordagem das NDNs evita que sejam necessários endereços dos nós (IP) e também que sejam necessários endereços MAC para identificar cada um dos nós. No fundo, tal como acontece no protocolo BOND, anteriormente abordado, a importância é dada aos dados e à forma como são identificados.

Desta forma, o protocolo LFBL pode lidar com dois tipos de mobilidade. Em primeiro lugar consegue lidar com a mobilidade física dos nós. Isto é, tem a capacidade de manter o encaminhamento dos dados independentemente da topologia da rede e da frequência com que esta se altera ao longo do tempo. Em segundo lugar, consegue lidar com a mobilidade lógica dos conteúdos. Isto porque, determinada informação poderá ser armazenada em qualquer nó, pelo que nenhum pedido deverá estar endereçado a um nó específico. As necessidades de um consumidor podem ser satisfeitas por qualquer um dos nós da rede, desde que possua os dados pretendidos [37].

Este protocolo opera sob quatro princípios básicos: manter o mínimo de informação acerca do estado da rede; transmitir tudo e assimilar o que escuta; tomar por si só todas as decisões de encaminhamento; e usar nomes em vez de endereços.

Uma vez que se trata de uma rede móvel, a topologia da rede pode alterar-se com bastante frequência. Deste modo, qualquer informação acerca do estado da rede que os nós guardem estará rapidamente desatualizada.

Um outro principio segundo o qual este protocolo opera é: transmitir tudo e assimilar aquilo que escuta. Por forma a não depender de endereço MAC, este protocolo transmite todos os pacotes em modo *broadcast*, isto é, de um para muitos. Através do cabeçalho que incorpora nos pacotes, o LFBL transmite informações de controlo da rede. Deste modo, nunca são criados pacotes de controlo.

O terceiro princípio pelo qual este protocolo se rege é o facto de todas as decisões de encaminhamento serem tomados pelo nó que recebe o pacote. Ou seja, quando um nó recebe um pacote deve então tomar decisões acerca do encaminhamento do mesmo, sem que haja qualquer tipo de coordenação com os nós vizinhos.

Por fim, o último princípio é o facto de não utilizar endereços IPs, mas sim dados nomeados [37].

Em termos de encaminhamento, o LFBL tem um funcionamento em tudo similar com o protocolo BOND, abordado anteriormente. A maior diferença entre estes dois protocolos é o facto de no LFBL não se utilizar qualquer tipo de identificação dos nós, enquanto no BOND é utilizado o endereço MAC para proceder a essa mesma identificação. De resto, o funcionamento é praticamente o mesmo.

Um nó que está interessado em determinado conteúdo gera um pacote de interesse e caso ainda não tenha conhecimento acerca de um produtor, transmite-o para todos os nós que estiverem ao seu alcance. Caso seja conhecido o produtor dessa informação, o encaminhamento é feito tendo em conta a distância para o destino (ou seja, o produtor). Esta distância é determinada segundo uma métrica definida anteriormente, tal como acontece no protocolo BOND. Antes de encaminhar qualquer pacote, os nós ficam à espera de que um nó mais próximo do destino transmita aquele pacote. Quando se processa no sentido inverso, os nós utilizam também a distância ao destino de forma a encaminhar a informação o mais rapidamente possível.

4.2.4. Information-Centric Delay Tolerant Network – ICDTN

O Information-Centric Delay Tolerant Network [4] tenta conjugar as DTNs com as redes centradas na informação, das quais a NDN faz parte. Segundo os autores, o facto de as redes serem capazes de lidar com grandes atrasos tem-se tornado cada vez mais importante. Isto porque, têm aumentado a quantidade e diversidade de redes móveis em que facilmente se perdem as ligações. Sendo assim, todos os designs para a Internet do Futuro devem contemplar a possibilidade de grandes atrasos e frequentes desconexões.

Através da análise das características principais dos dois tipos de redes, os autores definem quatro pontos comuns a ambas: armazenamento de dados na rede, conceito de *late biding*, longevidade dos dados e encaminhamento flexível [4]. Partindo destes pontos em comum foi desenvolvido um novo design que sobrepõem os dois tipos de rede.

Posto isto, o ICDTN comtempla também ele dois tipos de pacotes denominados de “Objeto de Informação” (Pacote de Dados, nas NDNs) e “Pedido de Informação” (Pacote de Interesse, nas NDNs). Ambos os tipos podem manter-se na rede através do mecanismo *story-carry-and-*

forward presente nas DTNs, isto enquanto o seu *Time To Live*[17] (TTL) não for excedido. Quando um nó demonstra interesse num determinado tipo de informação, um Pedido de Informação (PI) é gerado. É então selecionado um conjunto de ligações ativas naquele momento, tendo em vista o encaminhamento do PI. O encaminhamento é feito utilizando uma estratégia de inundação. Cada um dos nós recebe uma cópia do PI. Uma vez recebido o PI, o nó calcula um grau de aceitação tendo em conta os Objetos de Informação (OI) guardados localmente. Se o PI for aceite, o processo de reencaminhamento por inundação recomeça. Assim que se atinge um produtor dessa informação, o OI percorre o mesmo caminho no sentido inverso [4].

4.2.5. Neighborhood-aware Interest Forwarding – NAIF

Considerando os desafios colocados pelas MANET, nomeadamente a elevada mobilidade dos nós e a conectividade intermitente, foi desenvolvido um protocolo de encaminhamento denominado de NAIF (*Neighborhood-aware Interest Forwarding*, [38]).

No método de encaminhamento NDNF (*Named Data Network Forwarding*, [25]) padrão (descrito na subsecção 3.2.) os pacotes de interesse são encaminhados de forma *broadcast*. São enviadas cópias para mais que um nó, aumentando assim o risco de congestionamento da rede. Por outro lado, o modo de funcionamento conduz ao desperdício de largura de banda. Há uma certa quantidade de dados que são difundidos sem qualquer utilização. Tendo em conta este problema, surge então o protocolo NAIF. Este protocolo surge como uma melhoria do protocolo NDNF, nomeadamente na fase de distribuição dos pacotes interesse.

Desta feita, os nós encaminham os pacotes interesse tendo em conta as suas estatísticas de encaminhamento. Essa estatística é utilizada para ajustar a taxa de encaminhamento. A taxa de encaminhamento indica a quantidade de pacotes de um determinado prefixo que devem ser encaminhados. Tendo por base a taxa de encaminhamento, um encaminhador decide encaminhar ou descartar um interesse.

Os nós têm em conta os pacotes de dados que circulam à sua volta, e, quantos mais pacotes com determinado prefixo o nó detetar, mais pacotes interesse com o mesmo prefixo o nó poderá descartar. A taxa de encaminhamento é reduzida. Isto porque, pacote de interesse descartado será, muito provavelmente, satisfeito por um dos seus vizinhos, não sendo necessário que o interesse seja encaminhado. Por outro lado, assim que um nó deteta que tem descartado muitos pacotes de interesse com determinado prefixo (isto é, deteta poucos pacotes

[17] Em português: "Tempo de Vida"

de dados com aquele prefixo na sua vizinhança), a taxa de encaminhamento é aumentada por forma a compensar. O nó encaminha mais pacotes de interesse com aquele prefixo [38]. Desta forma, evita-se que a rede fique congestionada.

4.2.6. Content-Centric Dissemination Algorithm for Delay-Tolerant Networks – CEDO

Hoje em dia, a mobilidade dos nós que compõem as redes coloca inúmeros novos desafios que tornam os protocolos existentes ultrapassados. A sua maioria suporta mal a mobilidade, não foram concebidos para responder a esse desafio. Por entre os novos desafios que surgem neste tipo de redes encontra-se a própria mobilidade dos nós, atrasos elevados e o facto de poderem não existir ligações entre determinados nós. Para responder a estes desafios, surgiram novas arquiteturas, novos protocolos e novos algoritmos capazes de lidar com a maioria destes problemas de uma forma bastante eficiente. Contudo, surgem novos problemas e é necessário o aprimoramento destas novas arquiteturas. Nomeadamente, desafios relacionados com segurança, estratégias de encaminhamento ou gestão de *buffers*.

Com o objetivo de melhorar as soluções até então descobertas e responder a estes desafios, foi criado um novo algoritmo de disseminação de dados denominado de CEDO (*Content-Centric Dissemination Algorithm for Delay-Tolerant Networks*, [39]). Por forma a lidar com os problemas de gestão de *buffers* e de agenda, isto é, determinar qual o pacote que deve ser encaminhado primeiro, é atribuído a cada um dos pacotes um *rating* de utilidade. Desta forma, os mais populares são encaminhados primeiro e os menos populares são descartados, caso o *buffer* se encontre cheio e seja necessário guardar um novo pacote.

Por forma a calcular esse nível de popularidade de um determinado conteúdo é necessário determinar a taxa de entrega desse mesmo conteúdo. Ou seja, é necessário determinar quantos dos interesses em determinado conteúdo são satisfeitos. Para tal é utilizada a fórmula:

$$DR^{(i)} = p^{(i)} \cdot q^{(i)} \quad (1)$$

Onde, $p^{(i)}$ indica a probabilidade de um nó receber uma cópia do conteúdo *i*; $q^{(i)}$ indica a taxa de pedidos do conteúdo *i*, ou seja, o quão determinado conteúdo é requisitado. Sendo assim, quanto maior for a taxa de entrega de determinado conteúdo, maior será a sua popularidade. Isto porque, uma maior taxa de entrega significa um maior interesse nesse conteúdo.

O grande objetivo deste algoritmo é aumentar a taxa de entrega de todos os conteúdos. Para otimizar esta solução por forma a atingir este objetivo, é fundamental que o número total

de cópias (de todos os conteúdos) não ultrapasse a quantidade de armazenamento de todos os nós, que não existam cópias duplicadas no mesmo nó e que exista pelo menos uma cópia de determinado conteúdo a circular na rede [39]. Apesar destas premissas terem que ser cumpridas, esta abordagem favorecerá os conteúdos mais populares, podendo mesmo extinguir os mais impopulares. É então necessário arranjar um equilíbrio para que tal não aconteça e a distribuição do armazenamento seja mais equilibrada.

Contudo, nas DTNs não é possível aceder a todos os *buffers* de todos os nós em simultâneo. Sendo assim, a cada novo contacto os nós terão de localmente decidir quais os pacotes que devem encaminhar primeiro e quais os que devem descartar se o outro nó tiver novos conteúdos e o seu *buffer* estiver cheio. Por forma a resolver este problema, é determinada a utilidade de cada um dos conteúdos. Tendo em conta a taxa de entrega de cada conteúdo abordada anteriormente, é calculada a utilidade desse determinado conteúdo recorrendo a uma fórmula matemática. Através dessa fórmula é possível verificar que com o aumento do número de cópias na rede, cada uma delas tende a ser menos útil. Isto é, quando o número de cópias de determinado conteúdo tende para infinito, a sua utilidade tende para zero. Este facto permite que mesmo os pacotes mais populares sejam descartados a determinado momento, uma vez que são menos uteis e são estes os primeiros a serem descartados. Desta forma, é garantido então um equilíbrio, garantindo-se assim que mesmo os conteúdos menos populares têm onde ser armazenados. É então criado como que um ciclo: quando dois nós se encontram, são replicados os conteúdos mais populares e são descartados dos seus *buffers* os conteúdos menos populares. Ao serem replicados, a utilidade desse conteúdo baixa, como já vimos anteriormente e serão eliminados dos *buffers* num momento seguinte. Ao serem eliminados dos *buffers*, a sua utilidade aumenta pois existem menos cópias a circular na rede. E assim sucessivamente, garantindo que, eventualmente, todos os conteúdos serão replicados numa fase, e descartados noutra.

Esta nova descoberta levou a uma nova abordagem: a utilidade de determinado conteúdo é proporcional à taxa de falhas de entrega. Essa taxa pode ser determinada pela seguinte equação:

$$MR^{(i)} = q^{(i)} - DR^{(i)} \quad (2)$$

Isto significa que quanto maior for a taxa de falha de entrega de determinado conteúdo, maior será a utilidade desse conteúdo. Ao aumentar a sua utilidade, serão replicados mais pacotes e a taxa de entrega irá aumentar, fazendo com que hajam menos falhas de entrega.

Menos taxa de falhas de entrega significa que um conteúdo não é tão útil e sendo assim, serão replicados menos pacotes. Cria-se assim um ciclo.

Apesar de existir agora um equilíbrio, os dados utilizados para calcular quão útil é um conteúdo são dados globais: taxa de requisição e taxa de entrega. Contudo, como dito anteriormente, nem sempre é possível aceder a todos os nós ao mesmo tempo e ter conhecimento destes valores. Posto isto, é necessário proceder à estimação desses mesmos valores. É necessário estimar quão útil é um conteúdo.

Para fazer essa estimativa, cada nó guarda o tempo a que determinado conteúdo foi requerido e o tempo em que essa requisição foi respondida. É assim feita uma primeira estimativa. Se o conteúdo chegar rapidamente, significa que existem vários pacotes com esse conteúdo a circular na rede. Para refinar essa estimativa, o nó tem em conta os tempos guardados pelos seus vizinhos e os tempos guardados por si mesmo.

Capítulo 5 – Conceptualização do PIFPv2

O protocolo PIFP pretende lidar com os dados nomeados num ambiente oportunista. Para tal, baseia-se no protocolo de encaminhamento das redes DTN, o PRoPHET [10]. Tendo em conta o passado dos nós e os seus contactos, este protocolo calcula a probabilidade de dois nós se voltarem a encontrar. Quantas mais vezes dois nós se encontrarem mais provável é que se voltem a encontrar. Seguindo este conceito surgiu o protocolo PIFP. Apesar de ter em conta os encontros anteriores, este protocolo não calcula a probabilidade de dois nós se voltarem a encontrar, mas sim a probabilidade de determinado conteúdo contactar com interessados. O mesmo conceito pode ser aplicado aos dados, como se propõe neste trabalho.

5.1. Protocolo PIFPv1 – Descrição Geral

Em [8] foi sugerida uma abordagem para a implementação do protocolo PIFP. Em suma, a cada entrada na FIB fica também associada uma previsibilidade. Cada vez que dois nós se conectam trocam um resumo do conteúdo que possuem na CS. Consoante o resumo que cada um deles recebe atualizam o valor das previsibilidades. Esta traduz a probabilidade de o nó contactar com determinado conteúdo. Por fim, tendo em conta o valor de cada previsibilidade encaminham ou não os pacotes interesse. Ou seja, existe uma comparação entre as probabilidades que cada um dos nós tem em contactar determinado conteúdo. Caso o outro nó possua uma probabilidade maior, o pacote interesse correspondente àquele nome é encaminhado.

Por forma a obter informação mais completa acerca das probabilidades de encontrar conteúdos, foi também implementada uma atualização das mesmas de forma transitiva. Nesta fase os nós, para além de trocarem o resumo de conteúdos da CS, trocam também as informações da FIB. Este processo é denominado de processo de convergência. A troca de informações da FIB permite aos nós calcular a previsibilidade através da regra de transitividade. Ou seja, se determinado nó "A" tem uma elevada probabilidade de encontrar nós que possuem o conteúdo *c*, então "A" é um bom nó para encaminhar interesses nesse conteúdo.

Para além desta abordagem foi ainda implementada uma outra versão do protocolo PIFP. Foi implementada uma versão proactiva. Nesta outra versão os conteúdos são encaminhados mesmo que não seja recebido qualquer pacote interesse. O nó recebe esses conteúdos guarda-os na CS enquanto esta tiver espaço livre. Apesar de satisfazer mais interesses, esta versão aumenta imenso a carga na rede. Isto porque existem conteúdos que são transmitidos, mas que não são posteriormente utilizados. Ou seja, são transmitidos mas podem não satisfazer qualquer interesse que possa existir no futuro.

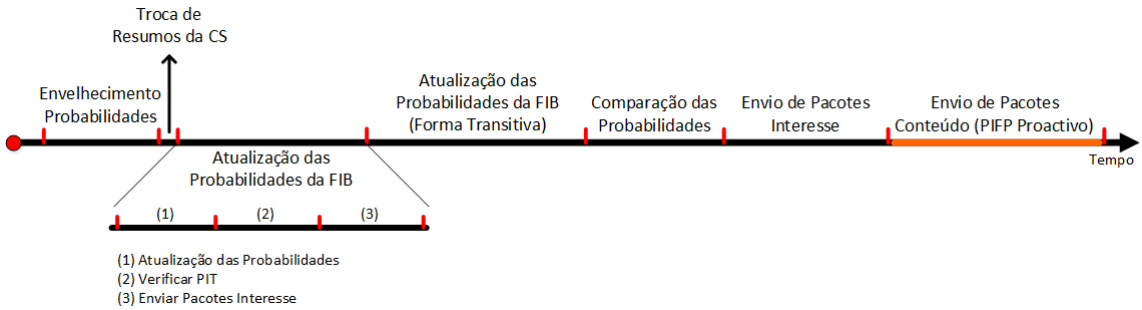


Figura 20 - Vista Cronológica do Protocolo PIFP

Através da Figura 20 é possível verificar o funcionamento do protocolo PIFP que havia sido implementado. Podem ver-se, ao longo do tempo, as diferentes funções que os nós executam assim que existe uma nova conexão. A parte final (destacada a cor-de-laranja) apenas acontece na versão proactiva do protocolo.

```
1.  Envelhecer Previsibilidades da FIB
2.  Quando receber resumo da CS
3.      Para cada entrada do resumo fazer
4.          Atualizar Probabilidade na FIB
5.          Verificar se consta na PIT
6.              Se constar, então
7.                  Envia Pacote Interesse
8.              Senão
9.                  Não fazer nada
10. Quando receber informação de encaminhamento
11.     Verificar se existe correspondência na CS
12.         Se existir, então
13.             Não fazer nada
14.         Se não existir
15.             Introduzir nova entrada na FIB
16. Para cada entrada da FIB fazer
17.     Verificar se existe correspondência na FIB do outro nó
18.         Se não existir
19.             Não fazer nada
20.         Se existir, então
21.             Se P(a,ci) < P(b,ci), então
22.                 Envia Pacote Interesse
23.             Senão
24.                 Não fazer nada
25.         fim se
```

Pseudo-código 1 – Resumo do Funcionamento do Protocolo PIFP

Através do Pseudo-código 1 pode ver-se, de forma sucinta, o funcionamento do protocolo PIFP antes das alterações efetuadas. Neste pseudo-código apenas estão descriminadas as tarefas principais. São executadas também outras tarefas que servem como auxílio às tarefas aqui representadas.

5.2. Protocolo PIFPv2 – Proposta de Alterações

Tendo em vista a resolução do problema criado pela abordagem proactiva decidiu-se desenvolver uma solução baseada no que já existia no protocolo PIFP inicialmente implementado. No protocolo original existe uma probabilidade de contacto associada a cada conteúdo. A solução apresentada passa por associar também uma probabilidade, mas a cada interesse. Até agora havia sido feito o cálculo de probabilidades para encaminhar interesses. Agora, será também feito um cálculo de uma probabilidade para encaminhar conteúdos.

O conceito segue o mesmo que já havia sido implementado anteriormente no sentido consumidor-produtor. Contudo, agora será implementado no sentido inverso, ou seja, no sentido produtor-consumidor.

Transportou-se a ideia original do protocolo PRoPHET, de que se dois nós se encontram a probabilidade de que se voltem a encontrar aumenta, para o conceito dos dados nomeados. Assim sendo, se um nó se cruza frequentemente com outros nós interessados no conteúdo "A", então esse nó é um bom nó para encaminhar mensagens com o conteúdo "A".

Quando dois nós se encontram, para além de trocarem um resumo do conteúdo da CS, trocam agora também o nome dos conteúdos que possuem na sua PIT. Tendo em conta esses nomes atualizam as suas probabilidades $P(a, i_c)$ de encontrar interessados, a qual traduz a possibilidade de o nó a encontrar interessados no conteúdo c.

Para chegar a estas probabilidades são utilizadas as mesmas equações que anteriormente eram utilizadas para calcular as probabilidades de contactar com determinado conteúdo. Com efeito, $P(a, i_c)$ aumenta sempre que o nó com o qual se está em contacto contém entradas na PIT correspondentes ao conteúdo c.

$$P(a, i_c) = P(a, i_c)_{antigo} + \left(1 + P(a, i_c)_{antigo}\right) \times P_{init} \qquad P_{init} \in [0,1] \qquad (3)$$

Quando determinado nó contacta pela primeira vez com um novo interesse, que até então não constava da sua lista de interesses, a variável $P(a, i_c)_{antigo}$ toma o valor de zero.

Já a variável P_{init} toma o valor de 0,75, pois é um fator de escala que fixa a que taxa se dá o aumento da probabilidade a cada contacto com determinado interesse.

Tal como acontece no PRoPHET, à medida que o tempo vai passando as probabilidades devem ir diminuindo. Isto significa que dois nós não se têm encontrado, e portanto, a probabilidade de se voltarem a encontrar vai diminuindo. O mesmo acontece neste conceito de dados nomeados. À medida que o tempo vai passando e um nó não contacta com outros nós interessados no conteúdo c, a probabilidade de voltar a encontrar interessados vai diminuindo.

$$P(a, i_c) = P(a, i_c)_{antigo} \times \gamma^k \qquad 0 < \gamma < 1 \qquad (4)$$

Na equação (4) γ é uma constante de envelhecimento. Quanto maior for este valor, mais rapidamente as probabilidades irão envelhecer. Segundo o protocolo PRoPHET a constante de envelhecimento deve ser igual a 0,98. Isto porque não são esperados contactos muito frequentes. Posto isto, também neste caso a constante γ assumirá o valor de 0,98. Por sua vez, k corresponde ao tempo decorrido (em segundos) desde que as probabilidades foram atualizadas pela última vez.

A primeira tarefa a ser executada é o envelhecimento das probabilidades. Utilizando a equação (4), as probabilidades existentes na lista de interesses ($L_i a$) são envelhecidas. A diminuição do seu valor é tanto maior quanto mais tempo um nó ficar sem se conectar com interesses num conteúdo. Para que a lista não fique muito extensa, as entradas e o correspondente conteúdo da CS cujo valor seja menor que um *threshold* são eliminadas. Se uma probabilidade for mais baixa que o *threshold* significa que o nó não tem contacto com interessados nesse conteúdo há bastante tempo. Assim, a entrada é eliminada prevenindo que a lista de interesses fique demasiadamente longa. Com uma lista mais curta, tanto a atualização das probabilidades como a posterior comparação das mesmas é feita de uma forma muito mais rápida. Por outro lado, ao eliminar o conteúdo da CS, evita-se que a capacidade de armazenamento se esgote. Uma vez que não têm existido contactos com interessados num conteúdo faz sentido apaga-lo da CS pois já não é útil.

Dando um exemplo mais prático, quando dois nós "A" e "B" se encontram, as probabilidades são envelhecidas. De seguida, "B" partilha a sua PIT com "A" e este copia os nomes de todas as entradas para uma lista. Essa lista é comparada com a lista de interesses ($L_i a$). Se já existir uma entrada com esse nome, então atualiza apenas a probabilidade. Caso não exista nenhuma entrada com aquele nome, então é registada uma nova entrada e é igualmente calculada a probabilidade de encontrar interessados naquele conteúdo.

Depois desta fase de atualização das probabilidades, é então invocado um outro método responsável por enviar os conteúdos. Assim sendo, "A" percorre todas as conexões que estão disponíveis no momento. Para cada uma delas, são percorridas todas as mensagens que "A" possui na CS. Se o outro nó já tiver essa entrada na sua CS, essa mensagem é passada à frente. Se não, é feita a comparação de probabilidades. O nó "A" percorre a sua lista de interesses ($L_i a$) até encontrar aquela entrada. Depois compara com a probabilidade expressa na lista de interesses do nó "B" ($L_i b$). Se a probabilidade do outro nó encontrar interessados for maior, a mensagem é adicionada a um *buffer* de saída. No final de percorrer todas as mensagens, as mensagens que se encontrarem no *buffer* de saída são organizadas por ordem de popularidade. Ou seja, as mais populares são enviadas em primeiro lugar. Só depois desta priorização é que as

mensagens são enviadas para a conexão em questão. Desta forma, se a conexão se perder, ficam entregues as mensagens que à partida têm mais interessados. Este processo realiza-se para cada uma das conexões disponíveis. Por sua vez, "B" aceita os conteúdos, mesmo que não exista nenhuma entrada na PIT com aquele conteúdo. Depois desta fase, "B" assim que encontrar outros nós poderá satisfazer os seus interesses.

```
1.  Aplicar equação 4
2.  Se P(a,ic) < threshold, então
3.      Elimina entrada da lista de interesses
4.  Quando receber nomes da PIT
5.      Para cada entrada da lista de nomes fazer
6.          Verifica se consta na Lia
7.              Se constar, então
8.                  P(a,ic)antigo ← Valor guardado
9.              Senão
10.                 P(a,ic)antigo ← 0.0
11.         Aplicar equação 3
12.         Adiciona a Lia
13. Para cada conexão disponível fazer
14.     Para cada mensagem fazer
15.         Se outro nó possui, então
16.             Continuar
17.         Senão
18.         Compara probabilidades
19.             Se P(a,ic) < P(b,ic), então
20.                 Adiciona à lista de saída
21.             Senão
22.                 Não faz nada
23.         Ordena lista de saída por popularidade
24.     Para cada entrada da lista de saída fazer
25.         enviar
```

Pseudo-código 2 – Resumo do Funcionamento das Novas Alterações do Protocolo PIFP

No Pseudo-código 2 está explanado o funcionamento das alterações feitas no protocolo PIFP. Tal como acontece no pseudo-código anterior, também neste está exposto um resumo das principais tarefas que são executadas. Existem outras tarefas que auxiliam estas principais, contudo não estão descriminadas no Pseudo-código 2.

Determinado nó, pode conter entradas na sua lista de interesses que não constam na sua PIT. Isto porque, tal como dito anteriormente, um interesse (e o seu consequente registo na PIT) só é enviado se o outro nó tiver mais probabilidade de encontrar determinado conteúdo. Porém, se um nó não tem muita probabilidade de encontrar aquele conteúdo, mas encontra com frequência interessados no mesmo, a sua probabilidade de encontrar interessados deve ser elevada. Mesmo não havendo nenhuma entrada na sua PIT. Assim, caso se cruze com um nó

que possua esse determinado conteúdo (num encontro esporádico), poderá receber esse conteúdo e satisfazer todos os interessados com os quais habitualmente se cruza.

Um dos objetivos principais da introdução deste componente é tentar diminuir a carga na rede. Tal como foi dito anteriormente, o protocolo PIFP Proactivo aumentava bastante a carga na rede. Eram transmitidos pacotes de dados sem que tal fosse pedido ou houvesse justificação. A única medida para enviar esses pacotes de dados era a popularidade de cada conteúdo. Os mais populares eram enviados primeiro. Contudo, não havia garantia que o nó que recebesse esses pacotes se cruzaria com nós interessados nesse conteúdo. Com este novo mecanismo, os pacotes de dados apenas são enviados caso o outro nó tenha uma maior probabilidade de encontrar interessados.

Para além desta redução da carga na rede espera-se também que existam mais interesses satisfeitos. Isto porque as CS de cada um dos nós incluirá apenas conteúdos para os quais contacta interessados com regularidade. Presume-se que todos os conteúdos presentes na CS servirão para satisfazer os interesses que surgem. Tanto os do próprio nó, como dos nós com os quais contacta frequentemente. Ao contrário do que acontecia no PIFP Proactivo, não existirá conteúdo potencialmente "inútil" a ocupar espaço na CS, permitindo assim um melhor aproveitamento dos recursos de cada um dos nós.

Por fim, o facto de se encaminhar conteúdos consoante a probabilidade de cada nó encontrar interessados para o mesmo permite que os interesses que surjam sejam satisfeitos mais rapidamente. Se um nó com o qual se costumam conectar armazenar conteúdo, é mais fácil para um nó interessado nesse conteúdo aceder ao mesmo.

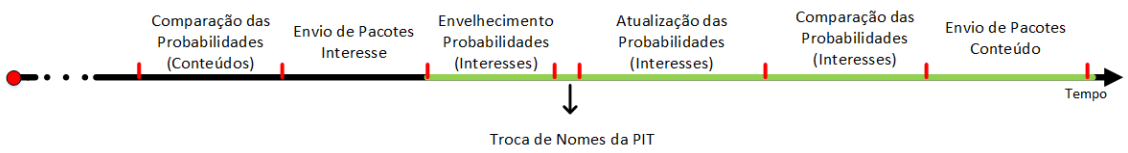


Figura 21 - Vista Cronológica do Protocolo PIFP com os Novos Componentes

Através da Figura 21 é possível ter uma visão cronológica do novo funcionamento do protocolo PIFP. Tal como se pode verificar, tudo que era processado anteriormente, continua a ser igualmente processado. A verde podem ver-se os novos mecanismos introduzidos no protocolo PIFP. Com as alterações feitas no protocolo a utilização do modo proactivo passa a não ter utilidade. Não só porque uns dos objetivos da nova implementação é eliminar os

problemas que o modo proactivo provocavam, mas também porque simplesmente não fazia sentido distribuir cópias de conteúdos pela rede indiscriminadamente.

Capítulo 6 – Plataforma de Simulação ICONE

Nos capítulos anteriores foi elaborado um estudo acerca das redes tolerantes a atrasos, acerca das redes de dados nomeados e acerca de trabalhos já feitos para aproximar estas duas redes. Uma vez que o objetivo deste projeto é aliar as NDNs com as DTNs é necessário um ambiente de simulação para que possam ser testadas as várias soluções por forma a atingir esse fim. Para tal, utiliza-se a plataforma de simulação ICONE (*Information-Centric Opportunistic Network Environment*).

O ICONE não é mais que um conjunto de novas características adicionadas ao The ONE [7]. Este é um simulador de DTNs, implementado em JAVA. O The ONE foi desenvolvido tendo em conta o paradigma das DTNs, ou seja, os nós e as suas posições. Neste simulador as mensagens são entregues tendo em conta o endereço da origem e do destino. Por forma a poder testar até que ponto as NDNs são aplicáveis em DTNs foi necessário proceder a algumas alterações na arquitetura deste simulador. Desta forma, as mensagens passaram a ser encaminhadas segundo o seu conteúdo e não os endereços de origem e destino.

6.1. Simulador "*The ONE*"

Como referido anteriormente, as DTNs sempre foram alvo de grande estudo. Não só pela sua versatilidade, mas também pela crescente necessidade de fazer chegar informação a áreas remotas. Com o intuito de analisar o comportamento dos diferentes protocolos de encaminhamento desenvolvidos para DTNs, e até mesmo testar aplicações, surgiu o simulador The ONE.

Este simulador possui um conjunto de ferramentas que permitem criar cenários reais, ou bastante parecidos com a realidade. Com um conjunto de parâmetros facilmente programáveis, os diferentes protocolos podem ser testados e avaliados em diferentes situações [7].

Possibilita que a mobilidade dos nós seja diferente a cada simulação, sendo que suporta vários modelos de mobilidade. Estes diferentes tipos de mobilidade apresentados pelos nós

permitem obter resultados mais realistas. Isto porque a forma como os nós se movem influencia a frequência, a duração e outros parâmetros dos possíveis encontros entre os nós.

Para além de suportar a mobilidade dos nós, o ONE suporta também geração e troca de mensagens, vários protocolos de encaminhamento e aplicações DTN, noções de consumo de energia, visualizar os nós em movimento e possui ainda interface para importar e exportar rotas de mobilidade, eventos e mensagens.

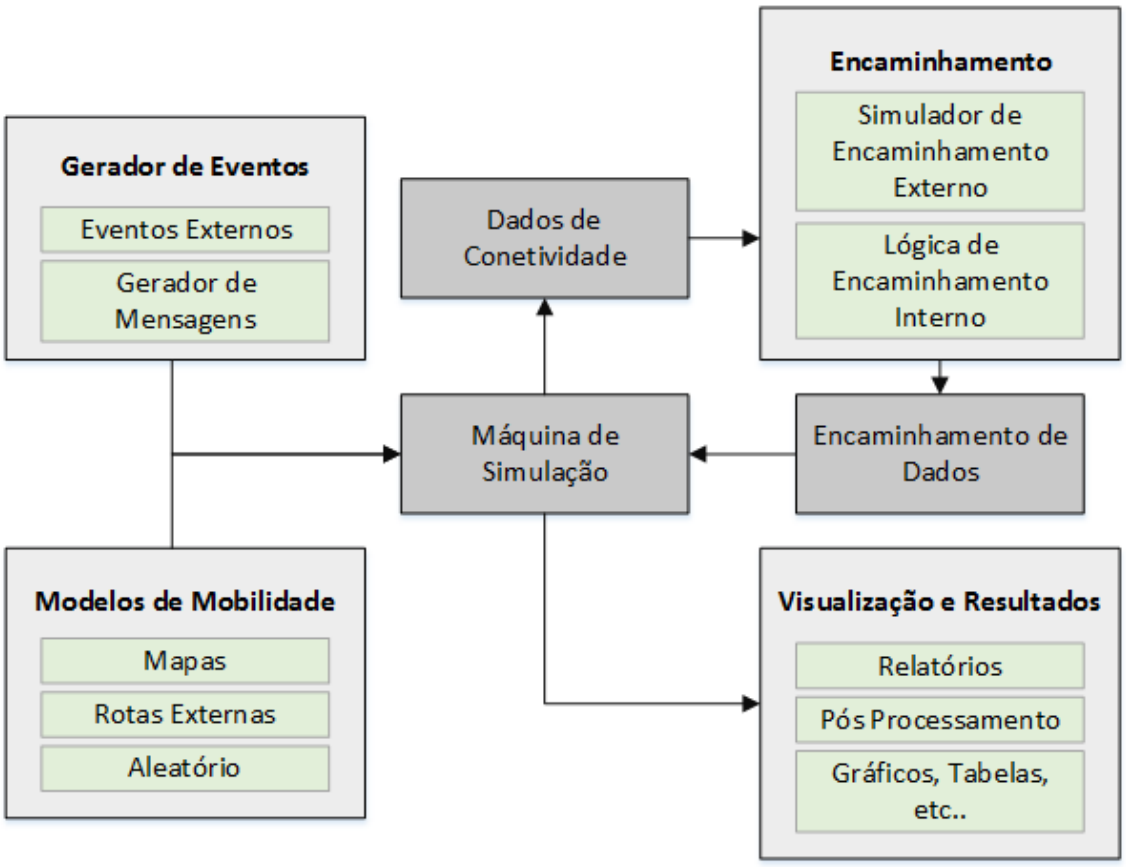


Figura 22 - Arquitetura do Simulador "*The ONE*" (adaptado de [7])

Através da Figura 22 é possível verificar a arquitetura deste simulador. Pode ver-se como os diferentes componentes interagem entre si.

A mobilidade dos nós é implementada pelos modelos de mobilidade. É no componente "Modelos de Mobilidade" que existem as ferramentas necessárias para atribuir essa mobilidade aos nós. A conectividade entre os nós é baseada na sua localização, alcance de comunicação e velocidade de transferência. O encaminhamento das mensagens é definido pelos módulos adequados que decidem quais as mensagens que devem ser reencaminhadas a cada novo contacto. Por fim, as mensagens são criadas pelo gerador de eventos, sendo que, estas são sempre *unicast*, existem apenas um destino e uma fonte para cada uma das mensagens.

Para cada simulação é definido um conjunto de parâmetros: tempo de simulação, modelo de movimento, alcance de conexão, velocidade de transmissão, protocolo de encaminhamento, etc… Para se alterar estes parâmetros basta aceder a um ficheiro de configuração. Este facto torna o simulador bastante flexível podendo simular vários cenários completamente diferentes.

Existem também ferramentas que permitem recolher os resultados das simulações. Esses resultados estão contidos em relatórios produzidos pelo componente "Visualização e Resultados". Alguns dos relatórios são produzidos durante a simulação, outros são produzidos apenas no final. Por outro lado, o ambiente gráfico oferecido pelo ONE permite visualizar o estado da simulação, mostrando a posição de cada um dos nós, quais os nós ativos e as mensagens que carregam [7].

6.2. Plataforma ICONE

Tal como referido anteriormente, para que seja possível simular o comportamento de uma rede centrada na informação e todos os seus protocolos, é necessário desenvolver uma plataforma adequada. Para tal, estendeu-se o simulador de DTNs "*The ONE*". Ou seja, foram implementados novos módulos, novas funcionalidades por forma a suportar o novo paradigma centrado na informação. Desta feita, o simulador ONE, descrito anteriormente, recebeu vários *upgrades* que lhe permitem agora simular protocolos centrados na informação, suportando igualmente a mobilidade dos nós e todas as outras características presentes nas DTNs.

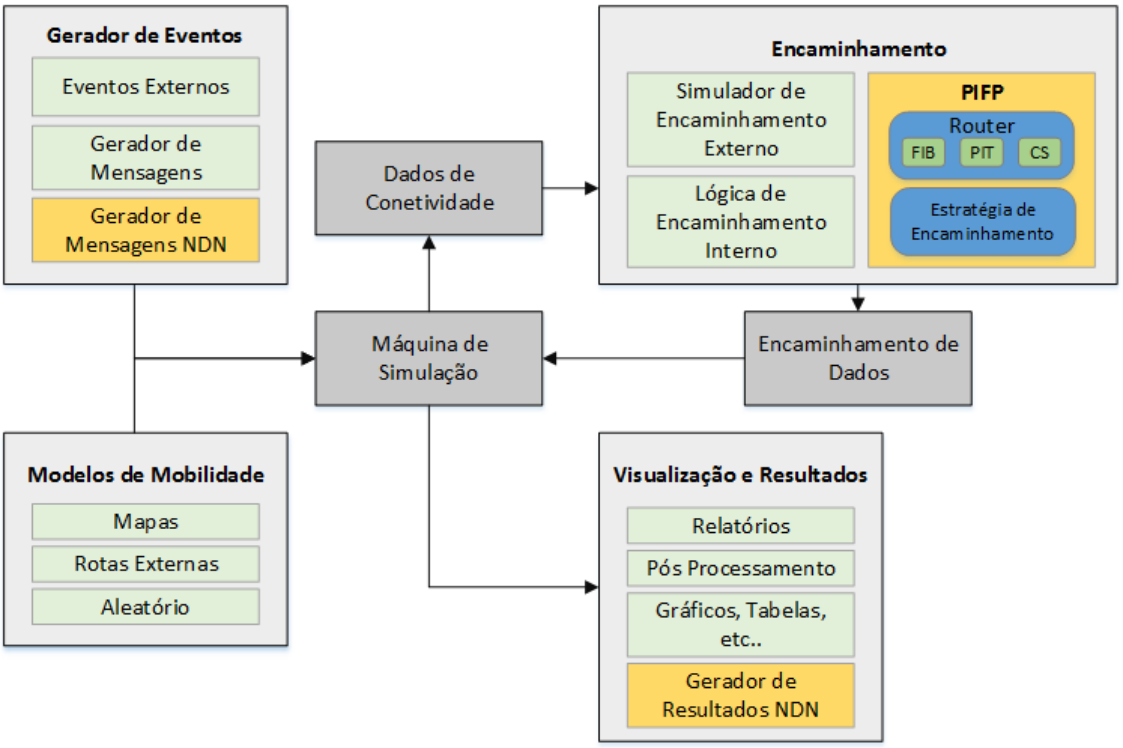


Figura 23 - Arquitetura Plataforma ICONE (adaptado de [8])

Através da Figura 23 pode ver-se a arquitetura da plataforma ICONE. Tendo como base a Figura 22, que representa a arquitetura simulador ONE, pode ver-se a amarelo os novos componentes NDN introduzidos.

Foram introduzidos vários componentes NDN, nomeadamente: *router* NDN, Protocolo PIFP, Gerador de mensagens NDN, Mensagens NDN e Gerador de Resultados NDN.

O módulo "*Router* NDN" implementa os módulos básicos das redes NDN presentes em cada encaminhador. Ou seja, a cada um dos nós são adicionadas as tabelas PIT e FIB, bem como a CS. Para além disso, é neste componente que se procede à manutenção das mensagens armazenadas. Por exemplo, cada entrada na PIT possui um TTL. Quando este expira, a entrada deve ser apagada libertando espaço no *buffer*. Contudo, existe um mecanismo de relaxamento do TTL. Ou seja, mesmo que o TTL chegue ao fim a entrada não é apagada imediatamente. Devido aos grandes atrasos característicos das DTNs, o TTL pode expirar e o pacote interesse pode ainda nem ter chegado ao produtor do conteúdo desejado. Este mecanismo aumenta o tempo no qual um interesse não é descartado.

Por sua vez, no "Protocolo PIFP" é implementado um protocolo de encaminhamento para as NDNs. É também o elemento responsável pela interação entre as camadas superiores e inferiores. Este componente será abordado mais adiante nesta dissertação.

No componente “Gerador de mensagens NDN” é gerado um conjunto de interesses seguindo uma filosofia semelhante à navegação web. Com isto aumenta-se substancialmente o realismo da simulação.

No módulo “Mensagens NDN” é definido um novo tipo de mensagem. Para que se possa introduzir um paradigma centrado na informação as mensagens a circular na rede terão de possuir uma outra estrutura diferente da que até então possuíam. Este componente é responsável por implementar esta nova estrutura nas mensagens.

Por fim, no componente “Gerador de Resultados NDN” são recolhidos os dados da simulação. Tal como acontecia anteriormente no simulador ONE, estes resultados podem ser recolhidos durante a simulação ou no final da mesma. Servem para avaliar o comportamento dos protocolos testados.

6.2.1. Protocolo PIFP

Por forma a garantir o melhor desempenho foi necessário implementar também um protocolo de encaminhamento capaz de lidar com este novo paradigma. Para tal, foi implementado no ICONE o protocolo PIFP. Este novo módulo é uma extensão do “NDNRouter”. Herda assim todas as suas estruturas de dados e todas as suas funções.

Para além de se tomarem as decisões de encaminhamento, definindo quais os pacotes a enviar, em que momento e para que conexão, é também neste módulo que se procede à atualização das previsibilidades.

Desta feita, assim que uma nova conexão é estabelecida o método “updateDeliveryPrevFor (DTNHost)” é invocado. É atribuído um argumento do tipo *DTNHost* a este método. Este argumento corresponde ao encaminhador com o qual se está conectado.

Primeiramente é trocado um resumo do conteúdo que cada um dos nós possui na sua CS. Tendo acesso a esse resumo é verificada a PIT. Caso existam entradas na PIT que possam ser satisfeitas, ou seja, caso existam interesses pendentes que podem ser satisfeitos, é enviado imediatamente um pacote interesse para esse nó. Aguarda-se assim que o outro nó receba esse pacote interesse e responda com o pacote conteúdo correspondente.

De seguida, e tendo em conta o resumo recebido, é calculada a previsibilidade de quão frequente é o contacto com determinado conteúdo. Por fim, tendo em conta este resultado são encaminhados os pacotes interesse correspondentes. Isto é, se determinado nó tem uma

elevada probabilidade de contactar com determinado conteúdo, é-lhe encaminhado o pacote interesse correspondente.

Para além de decidir para que nós enviar os pacotes interesse, baseado nas previsibilidades calculadas anteriormente, este módulo é também responsável por lidar com a receção dos pacotes. Assim que um pacote é recebido por determinado nó, este módulo verifica se se trata de um pacote interesse ou de um pacote de dados e lida com eles de forma diferente.

Caso se trate de um pacote de dados é invocado o método "onData (Message, String, DTNHost)", ao qual são atribuídos três argumentos. Tipo *Message* que corresponde à mensagem recebida, outro do tipo *String* que indica o nome da mensagem recebida e um último do tipo *DTNHost* que corresponde ao encaminhador que enviou a mensagem.

Este método começa por verificar a PIT, por forma a detetar se o pacote de dados recebidos satisfaz alguma das entradas nesta tabela. Em caso afirmativo é recolhido o id de todos os nós interessados naquele conteúdo. De seguida é verificado se o ID do nó que recebeu o pacote de dados consta como sendo um dos interessados. Se constar, significa que é um subscritor. Posto isto, uma cópia desta mensagem é guardada na CS e no repositório. É então reportado o sucesso no que toca à satisfação de um interesse. Caso não existam mais interessados neste conteúdo, a mensagem de interesse correspondente ao conteúdo recebido é eliminada. Para o caso de o nó não ser um nó subscritor, ou seja, se o nó não tiver interessado neste conteúdo, mas ainda assim tiver uma entrada na PIT (recebeu anteriormente pacotes interesse de outros nós) é guardada uma cópia da mensagem na CS mas não no repositório e é eliminada a entrada na PIT.

Caso se trate de um pacote interesse é invocado o método "onInterest (Message, String, DTNHost)". Tal como acontece com o método descrito anteriormente, são atribuídos os mesmos três argumentos.

Desta feita, o primeiro passo deste método é verificar se existe conteúdo armazenado na CS ou no repositório que possa satisfazer o pacote interesse recebido. Em caso afirmativo verifica se está no *buffer* de mensagens do nó. Se este não contiver o conteúdo desejado, verifica se existe esse conteúdo no repositório. Em caso afirmativo, copia a mensagem e envia a cópia. Em caso negativo é porque o conteúdo está no *buffer*. Sendo assim, percorre todo o *buffer* de mensagens até encontrar a mensagem com o ID pretendido. Encontrada essa mensagem, cria uma cópia e envia a cópia.

Caso o primeiro passo dê resultado negativo, ou seja, caso não exista conteúdo armazenado que possa satisfazer o interesse recebido é verificada a FIB. Esta verificação é feita por forma a perceber se existe informação acerca do conteúdo pretendido, nomeadamente, se alguma vez

foi feito algum contacto com esse conteúdo. Em caso afirmativo, o pacote interesse é guardado para que seja posteriormente enviado a um nó adequado. Em caso negativo, caso não exista qualquer informação acerca do conteúdo pretendido, o pacote interesse é descartado.

Para além de implementar estes métodos de gestão de probabilidades e gestão de pacotes recebidos este módulo implementa ainda outros métodos. Implementa, entre outro, métodos responsáveis por tentar enviar mensagens para outros nós e pela gestão e remoção de mensagens, nomeadamente interesses já satisfeitos.

6.2.3. Bloom Filters

Tal como referido anteriormente, nas redes tolerantes a atrasos os contactos entre os nós podem não existir, ou quando existem podem ser bastante curtos. Para além desta limitação, existe também o facto de a velocidade de comunicação entre os nós poder ser bastante baixa.

Tendo estes fatores em conta, é, portanto, bastante importante que as pesquisas nas diferentes tabelas e posteriores atualizações das probabilidades sejam feitas de uma forma rápida.

O facto de nas NDNs o endereçamento ser feito tendo em conta os nomes dos pacotes e não um endereço IP pode tornar esse processo um tanto ao quanto lento. Isto porque os nomes podem ter tamanhos diferentes e uma pesquisa por nome pode levar a que sejam percorridos um número enorme de caracteres até ser encontrado o nome procurado. Por outro lado, as tabelas onde é necessário efetuar as pesquisas de nomes (PIT, FIB e CS) podem ter um elevado número de entradas, atrasando ainda mais o processo. Para além destas limitações existe ainda o facto destas mesmas tabelas sofrerem atualizações constantemente devido aos contactos que se perdem e outros que vão surgindo.

Para contornar este problema e tornar a pesquisa nas diferentes tabelas mais célere recorreu-se à implementação de *Bloom Filters* [40]. Esta forma de pesquisa reduz o consumo de memória em comparação com os elementos que são armazenados diretamente. Um *Bloom Filter* consiste numa sequência de bits que se encontram, primeiramente, a 0. Ao adicionar um elemento são aplicadas funções de *hash*, um *k* número de vezes, fazendo com que alguns bits do vetor se alterem para 1. Cada função resulta numa posição diferente de bits a 1 dentro do vetor de bits. Os *bloom Filters* padrão, não suportam a remoção de elementos, isto porque quando uma função de *hash* é aplicada não é possível reverter o processo.

Uma vez que neste cenário é necessário não só inserir como também remover elementos do conjunto teve de aplicar-se um subconjunto de *Bloom Filters*, os *Counting Bloom Filters* [32]. Estes permitem a inserção e remoção de valores do conjunto. Para tal recorrem a um pequeno contador que é incrementado quando um elemento é inserido e decrementado quando um elemento é removido.

Posto isto, e por forma a tornar o processo de procura mais rápido, foram aplicados *Counting Bloom Filters* às várias tabelas que constituem um nó NDN, nomeadamente: FIB, PIT e CS.

6.2.4. Algoritmos de Gestão da *Content Store* e *Repository*

Uma característica comum a estes dois tipos de redes é o facto de ambas armazenarem dados ao longo do caminho que os conteúdos percorrem até chegar ao destino.

O facto de se guardar informação ao longo do caminho percorrido permite uma resposta muito mais rápida caso seja necessária uma retransmissão. Para além disso, este armazenamento permite que nas DTNs os dados sejam encaminhados assim que existir um contacto, sendo que estes nem sempre estão disponíveis. Permite também que nas NDNs a resposta a interesses seja muito mais célere, uma vez que não é necessário pedir ao produtor para reenviar a mesma mensagem.

Desta feita, existe em cada um dos nós dois tipos de estruturas com a mesma finalidade, contudo com funções diferentes: *Content Strore* (CS) e *Repository* (ou *Repo*). A CS é uma estrutura que guarda os dados de uma forma persistente, isto é, os dados ficam armazenados nesta estrutura até que não exista mais espaço disponível. Assim que se atinge o limite de espaço disponível, alguma dessa informação é apagada tendo em conta a estratégia definida. As estratégias de gestão de *caching* serão discutidas mais adiante nesta dissertação.

Os *Repository* ou *Repo* são uma estrutura "extra" que permite aumentar a capacidade de resposta. São utilizados apenas para dar resposta a futuros pedidos da rede. Este esquema segue o mesmo conceito das *Content Delivery Networks* (CDN). Ou seja, é criada uma cópia de um determinado conteúdo já existente na rede e guardado na *Repo*. Desta forma os pedidos de conteúdo poderão ser satisfeitos mais rapidamente uma vez que não é necessário contactar o produtor.

Posto isto, foi implementado no ICONE o *Repo*. Tiveram de ser aplicadas algumas alterações, não só para permitir suporte a esta nova estrutura, como também para promover a

cooperação com as estruturas já existentes, nomeadamente a CS. Tal como referido anteriormente, os *Repos* permitem aumentar a capacidade de armazenamento dos nós. Contudo, numa rede oportunista nem todos os nós apresentam capacidade para guardar grandes quantidades de informação. Quer por escassez de armazenamento, processamento ou até mesmo por questões de limitação de energia. Dadas estas limitações, nem todos os nós terão esta estrutura "extra" associada.

Por outro lado, todos os nós possuirão uma CS. Esta trata-se de um *buffer* de rede no qual os dados são guardados de forma persistente. Isto significa que os conteúdos aqui armazenados serão utilizados para responder a falhas da rede, mas também para satisfazer futuros interesses que possam surgir. Esta estrutura tem uma capacidade de armazenamento reduzida e exigem menos capacidade de processamento quando comparada com os *Repos*.

Uma vez que a CS tende a ficar cheia com alguma facilidade é necessário proceder à eliminação de algum conteúdo lá armazenado. Para tal, torna-se bastante importante definir quais os conteúdos que devem ser apagados para libertar espaço permitindo armazenar conteúdos novos. Foram então testados, mas não implementados no ICONE, vários algoritmos de gestão da CS [9].

Foram então testados algoritmos básicos de gestão de *buffers*, como por exemplo o FIFO. O algoritmo *First-In-First-Out*[18] (FIFO) é um algoritmo bastante simples. O primeiro elemento a ser inserido na fila é também o primeiro a ser eliminado assim que se atinge o limite de espaço. O elemento que foi inserido há mais tempo é o primeiro a ser eliminado.

Outro algoritmo testado foi o *Least Recently Used*[19] (LRU). Este é também um algoritmo bastante básico. Desta vez, assim que a *cache* estiver cheia é eliminado o elemento que foi acedido há mais tempo.

Foi testado também o algoritmo *Least Frequently Used*[20] (LFU). Neste algoritmo, assim que a *cache* tiver cheia, é eliminado o elemento que foi acedido menos vezes. Ou seja, em vez de se utilizar o fator tempo para comparar os diferentes elementos, é utilizada a frequência com que cada elemento é acedido. Assim sendo, o que for acedido menos vezes é eliminado.

Foi ainda testado um algoritmo aleatório. Neste, quando se atinge o limite de capacidade da *cache* é escolhido um qualquer elemento aleatório para ser eliminado.

[18] Em português: "Primeiro a Entrar Primeiro a Sair"
[19] Em português: "Usado Menos Recentemente"
[20] Em português: "Usado Menos Frequentemente"

Por fim, foi testado um algoritmo que tem em conta os interesses da rede. Este algoritmo tem em conta os interesses que chegam a um determinado nó. Desta feita, existe uma lista de interesses, assim que um interesse é recebido é verificado se já consta da lista. Em caso afirmativo, é incrementado o número de pedidos; Em caso negativo, é adicionada uma nova entrada a essa lista. Tendo essa lista em conta, são eliminados da CS os conteúdos que tiverem o menor número de pedidos. Isto porque são os conteúdos que menos interessam aos nós vizinhos, são os conteúdos menos populares.

Tal como referido anteriormente, todos estes algoritmos foram testados e os seus resultados apresentados em [9], contudo nenhum foi implementado no ICONE. Desta feita, os conteúdos da CS apenas são eliminados quando o seu tempo de vida chega ao fim.

6.2.5. Geração de Mensagens

Uma vez que se pretende efetuar uma simulação por forma a verificar o comportamento do protocolo, a geração de mensagens é também ela simulada. No simulador The ONE esta geração pode ser feita de duas formas: ou através de uma aplicação programada para o efeito ou através da leitura de ficheiros externos. Neste caso, as mensagens são geradas com recurso a ficheiros externos.

Através destes ficheiros procurou-se desenvolver um ambiente de simulação mais realista. Para tal as mensagens de interesse são geradas tendo como base o tráfego HTTP. Posto isto, durante a navegação web são gerados pedidos HTTP que num ambiente de dados nomeados corresponde a pacotes de interesse.

Em cada sessão estabelecida existem vários novos pedidos (interesses) que surgem a cada *click* do utilizador.

Quando se inicia a simulação é carregado um ficheiro de texto no qual estão descritos os interesses. Ou seja, existe um ficheiro que indica o momento em que surge o interesse num conteúdo. Isto é, é lido um ficheiro que contém a informação acerca do momento em que determinado interesse surge, qual o nó que está interessado nesse conteúdo, o nome do conteúdo em si, entre outros parâmetros.

Tabela 2 - Exemplo de Ficheiro para Gerar Interesses

Tempo (s)	Comando	ID	Origem	Tamanho	Nome
900.280	C	M1	10	700	uminho.pt/di

É também lido um ficheiro no qual se carrega conteúdos para cada um dos nós produtores. Nas simulações que serão efetuadas existirão dois tipos de nós, os consumidores e os produtores. Neste segundo ficheiro é definido que produtor possuirá cada um dos conteúdos.

No que diz respeito ao tamanho das mensagens de dados, é atribuído um tamanho aleatório compreendido entre dois valores. Ou seja, tal como dito anteriormente o simulador The ONE é muito versátil e bastante configurável a cada simulação. Desta feita, pode escolher-se também qual o tamanho máximo e mínimo das mensagens. Cada vez que "surge" uma mensagem, o simulador atribui-lhe um tamanho aleatório compreendido entre os valores indicados no ficheiro com as configurações da simulação.

6.3. Implementação das Novas Funções do Protocolo PIFP

Assim que se dá um novo contacto é invocado o método "upadateDeliveryPredFor" descrito anteriormente. De seguida é invocado o método "updateTransitivePreds", também este descrito anteriormente. Com as alterações feitas no protocolo PIFP é agora invocado um terceiro método denominado de "updateIntProb (DTNHost)".

Este método agora implementado recebe apenas um argumento do tipo DTNHost. Este argumento indica qual o nó com o qual se está conectado. É através deste argumento que se consegue aceder às tabelas do outro nó (isto num ambiente de simulação) e é para o nó identificado por este argumento que serão enviadas as mensagens.

6.3.1. Envelhecimento das Probabilidades

Tal como foi referido no capítulo 5, a primeira tarefa a ser executada é o envelhecimento das probabilidades. O objetivo é ir atualizando as probabilidades de acordo com os contactos que surgem. Este envelhecimento traduz a ausência de contato. Quanto maior for o tempo decorrido desde o ultimo contacto, menores se tornarão as probabilidades. Posto isto, o método "updateIntProb" começa por invocar um outro método.

Para se envelhecer as probabilidades foi necessário implementar um novo método, denominado de "ageInterestProb()". Este método não recebe qualquer tipo de parâmetros. Inicialmente é determinado o tempo passado desde a última atualização. O tempo de quando

terminou o último envelhecimento é guardado numa variável. Obtém-se então o tempo atual e faz-se a diferença entre esse valor e o valor guardado como sendo o momento do último envelhecimento. Este resultado é guardado numa variável denominada de "timeDiff". Se este valor for igual a zero a execução deste método termina e o nó volta a executar o método "ageInterestProb". Tal significa que não decorreu tempo nenhum desde o último envelhecimento, portanto não faz sentido voltar a executar todo o processo se os resultados vão ser os mesmos.

Caso esta diferença seja diferente de zero, é executada a equação (4). Primeiramente é resolvida a potência γ^k, na qual k assume precisamente o valor da diferença de tempo, em segundos. O resultado deste cálculo é guardado na variável "mult".

Para cada entrada da lista de interesses é atualizada a probabilidade. Ou seja, é obtido o valor da probabilidade de cada uma das entradas e é multiplicado pela variável "mult". Se o resultado desta multiplicação for menor que o *threshold* é adicionada uma nova entrada na lista "deleteEntries". Esta é uma lista auxiliar, na qual se guardam os nomes das entradas $L_i a$ e dos conteúdos que devem ser eliminados. Caso o novo valor da probabilidade não ultrapasse esse *threshold* a entrada da lista de interesses é atualizada.

Por fim, é obtido e guardado o instante de tempo em que este processo foi concluído para que se possa determinar quanto tempo passou desde o último envelhecimento.

```
1.  tempoAtual ← obter tempo atual
2.  timeDiff = (tempoAtual – ultimoEnvelhecimento)
3.  k ← timeDiff
4.  mult = γ^k
5.  Para cada entrada de Lia fazer
6.      oldProb ← Obter probabilidade
7.      newProb = oldProb * mult
8.      Se newProb < threshold, então
9.          adiciona entrada na lista deleteEntries
10.     Senão
11.         atualiza entrada de Lia
12. Para cada entrada da lista deleteEntries fazer
13.     eliminar entrada da Lia
14.     elimina entrada da CS
15. lastAgeInterestUpdate ← obter tempo atual
```

Pseudo-código 3 - Envelhecimento das Probabilidades

No Pseudo-código 3 é descrito detalhadamente o modo como são envelhecidas as probabilidades existentes. Por forma a facilitar a compreensão da solução são utilizados os nomes das variáveis tal como na implementação.

6.3.2. Atualização das Probabilidades

Após o envelhecimento das probabilidades chega à altura de atualizar as mesmas tendo em conta o conteúdo da PIT do nó que se encontra do outro lado da conexão.

Obtém-se então o nome de todas as entradas da PIT do outro nó. Esses nomes são guardados numa lista de *String* denominada de "otherPIT". Caso a PIT do outro nó ainda não tenha qualquer entrada, a lista otherPIT estará vazia. Assim, o processo é interrompido uma vez que não existe contacto com interesses. Deve então garantir-se que a lista otherPIT tem de facto algum conteúdo. De seguida, essa lista é percorrida entrada a entrada e é verificado se esse nome se encontra na lista de interesses. Em caso afirmativo, o valor da probabilidade guardado na lista de interesses é copiado para a variável "oldProb". Em caso negativo, esta variável assume o valor zero.

Recorrendo à equação (3) é obtido o valor da probabilidade atualizada. O valor guardado na variável oldProb corresponde a $P(a, i_c)_{antigo}$ na equação. O resultado deste cálculo é guardado na variável "newProb".

Por fim, é introduzida na lista de interesses o nome e a nova probabilidade. Uma vez que a lista de interesses é do tipo HashMap a entrada anterior é apagada. Isto porque neste tipo de estrutura não é possível existir duas entradas com a mesma chave. Neste contexto, não é possível haver duas entradas com o mesmo nome.

```
1.  otherPIT ←obter nomes da PIT outro nó
2.  Se otherPIT vazia, então
3.      Retorna
4.  Senão
5.      Para cada entrada de otherPIT fazer
6.          Verifica se consta na Lia
7.          Se constar, então
8.              oldProb ← obter probabilidade
9.          Senão
10.             oldProb ←0,0
11.         newProb = oldProb + (1 – oldProb) * Pinit
12.         Adiciona entrada em Lia
```

Pseudo-código 4 - Atualização das Probabilidades dos Interesses

No Pseudo-código 4 é possível ter uma perceção da solução encontrada para atualizar as probabilidades. Neste pseudo-código são utilizados os mesmos nomes das variáveis utilizadas na implementação.

6.3.3. Comparação de Probabilidades e Envio das Mensagens Conteúdo

Após a atualização das probabilidades é chegado o momento de se proceder às comparações das probabilidades e de acordo com o resultado dessa comparação enviar os pacotes de dados adequados.

No final da atualização das probabilidades é invocado o método responsável por enviar as mensagens de dados, denominado de "sendContent()". Neste método é criada uma lista, denominada de "outgoingMessages", na qual estarão armazenadas as mensagens que devem ser enviadas para o outro nó.

Este método percorre todas as conexões disponíveis no momento. Para cada uma delas executa todo o processo que irá ser descrito de seguida.

Começa por verificar se a conexão em questão está disponível, ou seja, verifica se o outro nó ao qual está ligado está a transmitir. Caso esteja ocupado, passa à próxima conexão. Se estiver disponível, o processo segue com a mesma conexão. São percorridas todas as mensagens que o nó possui no seu *buffer* e é verificado se se trata de uma mensagem de dados ou uma mensagem de interesse. Apenas as mensagens de dados são processadas. Caso se trate de uma mensagem de interesse passa-se à mensagem seguinte. Assim que encontrar uma mensagem de dados, é verificado se o outro nó já possui essa mensagem, quer na sua CS, quer no seu *buffer*. Se o outro nó já possuir a mensagem, passa-se à mensagem seguinte.

Até este ponto fica então garantido que o outro nó está disponível para receber o conteúdo e que ainda não o possui. Chega então a altura de verificar se a sua probabilidade de encontrar interessados é superior. Para tal, recorre-se ao método "compareProb(NDNRouter, String)" e, dependendo do resultado retornado por este método, a mensagem é ou não adicionada à lista "outgoingMessages". Este método recebe como argumentos o ID do nó com o qual se está a comunicar e a o nome da mensagem que deve processada.

No final de se executar este processo para todas as mensagens existentes no nó, é verificado se a lista outgoingMessages está vazia. Caso esteja, a execução é interrompida. Caso não esteja vazia, a lista é organiza de acordo com a popularidade das mensagens que estão prestes a ser enviadas. Através do método "orderPopularity(List)", que recebe como argumentos a própria lista outgoingMessages, as mensagens são ordenadas com as mais populares a ficarem nas

primeiras posições da lista. Assim, espera-se que caso a ligação se perca a meio do processo, as mensagens que potencialmente irão satisfazer mais pedidos sejam enviadas primeiro.

Organizadas as mensagens, procede-se finalmente ao seu envio para o outro nó. É invocado o método "tryMessagesForConnected(List)", que recebe como argumento a lista outgoingMessages. Para que seja possível reportar o envio, sempre que uma mensagem é enviada com sucesso é adicionada uma entrada na lista "tupleSend".

```
1.  Para cada conexão fazer
2.      Verificar disponibilidade do outro nó
3.      Se estiver disponível, então
4.          Verificar se se Lib vazia
5.          Se estiver vazia, então
6.              Conexão seguinte
7.          Senão
8.              Para cada mensagem fazer
9.                  Verifica se é PC
10.                 Se for, então
11.                     Verifica se outro nó possui conteúdo
12.                     Se possuir, então
13.                         Próxima mensagem
14.                     Senão
15.                         Se compareProb() < 1, então
16.                             Verificar se lista outgoingMessages contém mensagem
17.                             Se contiver, então
18.                                 Próxima mensagem
19.                             Senão
20.                                 Adiciona entrada a outgoingMessages
21.             Verificar se outgoingMessage vazia
22.             Se vazia, então
23.                 Conexão seguinte
24.             Senão
25.                 outgoingMessages ← orderPopularity(outgoingMessages)
26.                 Envia mensagens
27.     Senão
28.             Conexão seguinte
```

Pseudo-código 5 – Envio das Mensagens Conteúdo

No Pseudo-código 5 está representado detalhadamente o processo de envio das mensagens de conteúdo. Tal como acontece nos pseudo-códigos anteriores, são utilizados os nomes das variáveis. Desta forma, espera-se que a implementação seja de mais fácil compreensão.

Tal como foi dito anteriormente o método "compareProb()" recebe dois argumentos. Um é o ID do nó com o qual se está a comunicar e o outro é o nome da mensagem a ser processada. É então verificado se existe na CS algum conteúdo com nome igual ao que foi atribui como

argumento. Se esta verificação for bem sucedida, o passo seguinte é verificar se existe na lista de interesses do outro encaminhador alguma entrada com o mesmo nome. Esta verificação é feita porque se não existem entradas com o mesmo nome em ambas as listas, significa que não existe comparação possível. Só faz sentido comparar probabilidades para o mesmo interesse. Caso exista a mesma entrada na lista de interesse de ambos os encaminhadores, é possível então proceder à comparação das probabilidades. Posto isto, a variável "interestProb" toma o valor de $P(a, i_c)$ (guardada na lista de interesses do encaminhador "A") e a variável "otherRouterProb" toma o valor de $P(b, i_c)$ (guardada na lista de interesses do encaminhador "B").

Por fim, se o valor de "interestProb" for menor que o valor de "otherRouterProb" deve ser enviado um pacote de conteúdo. Sendo assim, método "compareProb()" retorna 0 (sucesso). Caso alguma das verificações acima falhem, o método retorna 1.

```
1.  String name ← nome do interesse
2.  Verifica se "name" consta da CS
3.  Se constar, então
4.      Verifica se "name" consta de Lib
5.      Se constar, então
6.          interestProb ← obter P(a,ic)
7.          otherRouterProb ← obter P(b,ic)
8.          Se interestProb < otherRouterProb, então
9.              retorna 0
10. Senão
11.     retorna 1
```

Pseudo-código 6 - Comparação das Probabilidades

Através do Pseudo-código 6 é possível verificar de forma detalhada o processo de comparação das probabilidades de encontrar interessados. Tal como nos pseudo-códigos anteriores, foi utilizada uma linguagem próxima da linguagem de implementação, para que esta seja mais fácil de compreender.

Capítulo 7 – Resultados da Simulação

Neste capítulo apresenta-se um resumo dos testes feitos ao PIFP. Será apresentado e discutido um conjunto de resultados de simulação. O objetivo deste capítulo é demonstrar até que ponto as alterações feitas têm impacto na performance do protocolo PIFP.

Para além de simular o protocolo PIFP com as novas características será também simulado o mesmo protocolo porém sem essas alterações. Desta forma, tal como acontece em [8], será possível fazer uma análise comparativa de ambas as versões do protocolo PIFP.

Por uma questão de nomenclatura, a partir deste momento, o protocolo PIFP implementado em [8] será referido como "PIFPv1" e o protocolo PIFP com as novas características será denominado de "PIFPv2".

Ao determinar se um protocolo é ou não eficiente devem ser tidos em conta algumas métricas. Neste caso, para preceder a essa avaliação, serão utilizadas as seguintes métricas:

- Interesses Satisfeitos – Número de interesses satisfeitos, independentemente se foram satisfeitos por dados vindos da rede, ou por dados armazenados em cache.
- Latência – Tempo médio decorrido desde que um interesse é gerado até que é satisfeito.
- Sobrecarga de Mensagens – Quantidade de mensagens de dados e mensagens interesse a circular na rede.
- Energia Consumida – Média de energia gasta durante a simulação
- Número de Saltos – Número de nós percorridos ao longo do trajeto desde a fonte até ao destino.

Apresentados os parâmetros utilizados na avaliação do desempenho dos diferentes protocolos, resta agora perceber quais as configurações de simulação e quais os cenários onde as mesma irão ocorrer.

7.1. Configurações de Simulação

Tal como referido anteriormente, para que seja possível simular o comportamento do protocolo PIFP foi utilizado o simulador de DTNs "*The ONE*". Esta é uma ferramenta que pode ser facilmente estendida e assim alargar o leque de redes que suporta.

Assim sendo, é necessário definir em que condições ocorre a simulação. Nas simulações elaboradas ao longo desta dissertação foram utilizadas as mesmas configurações que em [8]. Desta forma é possível comparar resultados visto que ambos os testes foram feitos sob as mesmas condições.

Todos os protocolos serão simulados sob as mesmas condições. Ou seja, as configurações não serão alteradas consoante o protocolo a ser testado. Só desta forma é possível garantir uma comparação verdadeira do desempenho de cada um dos protocolos.

O cenário no qual serão simulados os protocolos corresponde ao mapa de Helsínquia, na Finlândia. Neste mapa estão incluídas estradas, linhas de elétrico, passeios e passadeiras.

Tabela 3 - Configurações de Simulação

Parâmetro	Valor
Tempo de Simulação	86400s (24horas)
Área de Simulação	4500m x 3500m
Velocidade do nó (m/s)	Pedestres – 0.5 a 1.5
	Carros – 2.7 a 13.9
	Elétricos – 7 a 10
Tempo de Vida das Mensagens	480 minutos (8 horas)
Interface de Rádio	Bluetooth
Alcance de Transmissão	Pedestres e Carros – 10m
	Elétricos e Pedestres Especiais – 1Km
Taxa de Transmissão	Interface Simples – 250Kbps
	Interface Alta Velocidade – 10Mbps
Capacidade de Armazenamento	Consumidores – 50MB
	Produtores – 100MB

Através da Tabela 3 é possível verificar quais as configurações da simulação.

Dependendo do grupo a que pertencem, os nós movem-se de acordo com modelos de mobilidade diferentes. Os nós descritos como pedestres movem-se a uma velocidade entre os 0,5 e os 1,5 m/s. Os carros deslocam-se a uma velocidade que varia entre os 2,7 e os 13,9 m/s. Por fim, os elétricos movem-se a uma velocidade de 7 a 10 m/s.

Existem quatro grupos distintos: pedestres, carros, pedestres especiais e elétricos. Dependendo do grupo a que pertence, um nó segue um modelo de mobilidade. Os pedestres (tantos normais como os especiais) seguem o modelo de mobilidade *Shortest Path Map-Based Movement*. Por sua vez, os nós que pertence ao grupo dos carros, utilizam o mesmo modelo de movimento, contudo, apenas podem circular nas estradas. Por fim, os elétricos utilizam o modelo de mobilidade *Routed Map-Based Movement*.

A interface de rádio utilizada é Bluetooth. Todos os nós utilizam a mesma interface de rádio. Contudo, existem alguns nós que possuem duas interfaces: uma interface básica e outra interface de alta velocidade. A interface básica tem uma taxa de transmissão de 200Kbps. Por outro lado, a interface de alta velocidade tem uma taxa de transmissão de 10Mbps.

Por fim, é feita uma distinção entre produtores e consumidores no que toca à capacidade de armazenamento. Os produtores têm uma capacidade de 100MB, já os consumidores têm capacidade para metade, ou seja, 50MB.

7.2. Cenários de Simulação

Para testar o desempenho da solução apresentada foram implementados dois cenários. Um primeiro cenário onde a densidade de nós é maior, ou seja, à partida existirá um maior número de contactos. E um outro cenário onde a densidade de nós é menor, o que à partida significa menor número de encontros entre os nós.

O primeiro cenário é composto por dois grupos de pedestres. O primeiro grupo tem trinta e dois elementos. Já o segundo grupo de pedestres é composto por trinta elementos. O que distingue estes dois grupos é o alcance de transmissão. Os elementos de um dos grupos, o de trinta e dois elementos, tem um alcance de transmissão de apenas dez metros e os do outro grupo têm um alcance de transmissão de um quilómetro. Este cenário é ainda constituído por um grupo de carros, com trinta e dois elementos e dois grupos de elétricos, com dois elementos cada. No total, este cenário é composto por noventa e oito nós. Por entre estes noventa e oito

nós existem oito que são produtores de conteúdos (4 elétricos e 4 pedestres), sendo todos os outros consumidores.

O segundo cenário, tal como o primeiro tem também dois grupos de pedestres. Contudo, desta vez um dos grupos é constituído por vinte e quatro elementos e o outro por apenas oito. Existe ainda um grupo de carros, de quinze elementos, e dois grupos de elétricos, com um elemento cada. Neste segundo cenário existem apenas quatro produtores de conteúdos, 2 elétricos e 2 pedestres.

É esperado que a performance no primeiro cenário apresentado seja superior ao segundo. Uma vez que a densidade dos nós é maior, irá haver mais conexões e, consequentemente, mais pacotes transmitidos. Por outro lado, a convergência dos nomes será mais rápida no primeiro cenário, devido ao maior número de conexões. Os atrasos na entrega de mensagens e a taxa satisfação de interesses também serão afetados pela diferente densidade de nós em cada um dos cenários. Espera-se que os atrasos sejam maiores no segundo cenário e que a taxa de satisfação seja maior no primeiro.

7.3. Apresentação e Discussão de Resultados

Por forma a perceber até que ponto as alterações introduzidas no protocolo PIFP trouxeram melhorias ao seu desempenho foram feitas várias simulações utilizando o ICONE. Serão apresentados os resultados das simulações feitas nos dois cenários descritos anteriormente. Desta forma, poderá perceber-se até que ponto a densidade da rede afeta o desempenho do protocolo. Serão também apresentados resultados de simulações utilizando vários tamanhos de mensagens. Por fim, serão expostos resultados no que toca ao número de saltos e de energia consumida.

7.3.1. Densidade da Rede

Neste teste foram utilizados dois cenários. Um cenário com um total de noventa e oito nós e o outro com uma total de 49 nós, de resto, descritos na subsecção 7.2. deste documento.

Objetivo principal destas simulações é perceber até que ponto a densidade e a mobilidade dos nós influencia o desempenho das duas versões do protocolo PIFP. Serão apresentado gráficos que demonstram os resultados obtidos, tanto a nível de interesses satisfeitos, como a nível de atraso na entrega das mensagens e ainda a nível de carga da rede.

As mensagens utilizadas nesta simulação têm um tamanho aleatório limitado entre 50k e 1M.

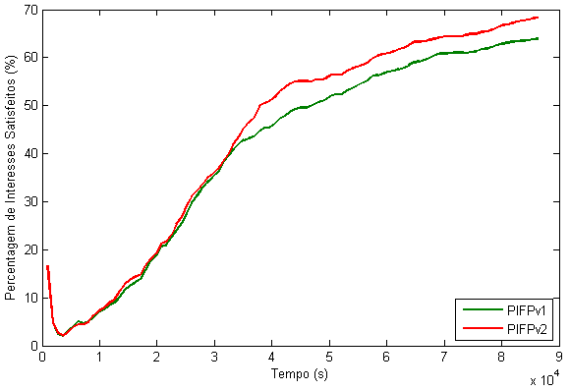


Figura 24 – Percentagem de Interesses Satisfeitos - Cenário 1

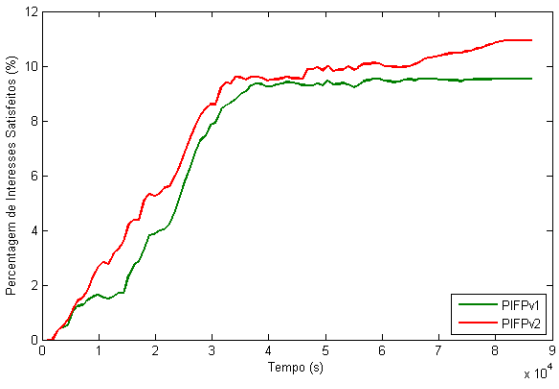


Figura 25 – Percentagem de Interesses Satisfeitos - Cenário 2

Tal como é possível verificar através das Figura 24 e Figura 25 o número de interesses satisfeitos baixa drasticamente com a baixa densidade de nós. Quanto menos nós existirem numa rede, mais escassos serão também os contactos entre eles. Os nós encontram-se menos vezes. O que se reflete na transmissão de dados entre os nós. O facto da comunicação entre os nós ser escassa, dificulta que as mensagens cheguem do consumidor ao produtor e vice-versa.

Por outro lado, o facto dos nós terem a capacidade de armazenamento pode levar a uma taxa de entrega superior. Como foi dito anteriormente, os nós têm a capacidade de armazenar conteúdos que mais tarde pode ser reutilizados se assim for necessário. Cada nó pode funcionar como uma *cache*. Ao existirem menos nós, esta distribuição de dados armazenados é também menor, o que se reflete na satisfação dos interesses.

No que diz respeito ao desempenho das duas versões do protocolo PIFP, pode ver-se que o PIFPv2 tem um desempenho ligeiramente superior em ambos os cenários. Tal acontece porque na versão 2 do PIFP existe uma maior distribuição de conteúdos. Os conteúdos são distribuídos tendo em conta a probabilidade de um nó encontrar interessados no mesmo. Ainda que um nó não demonstre um interesse direto em determinado conteúdo, se contactar com vários interessados, recebe uma cópia. Assim poderá satisfazer os interesses com os quais contacta habitualmente. Esta abordagem faz com que os conteúdos se espalhem pela rede de uma forma equilibrada, facilitando assim o acesso aos mesmos.

No cenário 2, um fator que pode ajudar a explicar a baixa satisfação de interesses do PIFPv2 é o facto de com a ausência de contactos, as probabilidades envelhecerem muito. As probabilidades são envelhecidas a cada novo contacto. Tendo em conta o tempo que passa

entre contactos, estas podem envelhecer bastante e passam o limite de 0.15. Ao baixarem deste limite, as entradas com as probabilidades são eliminadas, bem como os respetivos conteúdos da CS.

No que diz respeito ao cenário 1, o PIFPv1 satisfez 63% dos interesses totais. Já o PIFPv2 satisfez 66% dos interesses originados. É possível verificar uma ligeira melhoria no que toca à entrega de conteúdos.

Já no cenário 2, o PIFPv1 satisfez apenas 9% dos interesses. Enquanto o PIFPv2 satisfez 11%. Tanto uma versão como outra ficam muito aquém do esperado quando comparado com o cenário 1.

Através destes factos é possível confirmar que a densidade de nós tem um enorme impacto no desempenho do protocolo de encaminhamento.

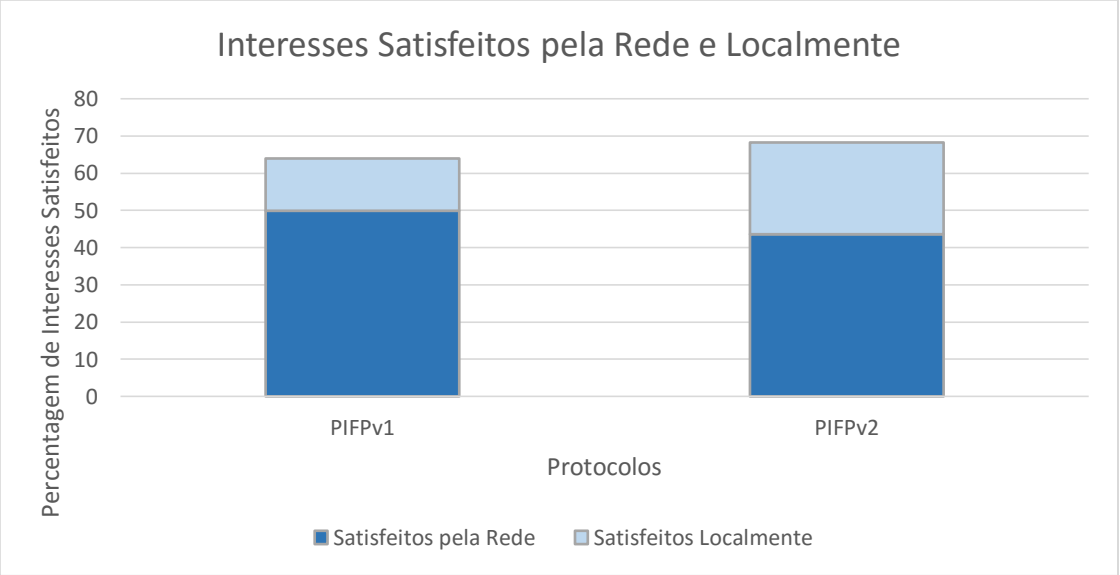


Figura 26 – Percentagem de Interesses Satisfeitos pela Rede e Localmente – Cenário 1

Através da Figura 26 é possível verificar a quantidade de interesses que são satisfeitos através de conteúdos recebidos pela rede e satisfeitos localmente, através de conteúdo armazenados na CS. Assim, na versão 1 do PIFP 78% dos interesses são satisfeitos através de conteúdo recebido pela rede e 22% são satisfeitos localmente, recorrendo a conteúdos armazenados na CS. Já no PIFPv2, 76% dos conteúdos são satisfeitos através da rede e 24% são satisfeitos localmente.

Este ligeiro aumento dos conteúdos satisfeitos localmente, no PIFPv2, deve-se ao facto de cada um dos nós armazenar mais conteúdos na sua CS. Ao receber conteúdos para satisfazer

outros nós, um nó poderá satisfazer um futuro interesse seu naquele conteúdo. Isto porque já o contém na sua CS.

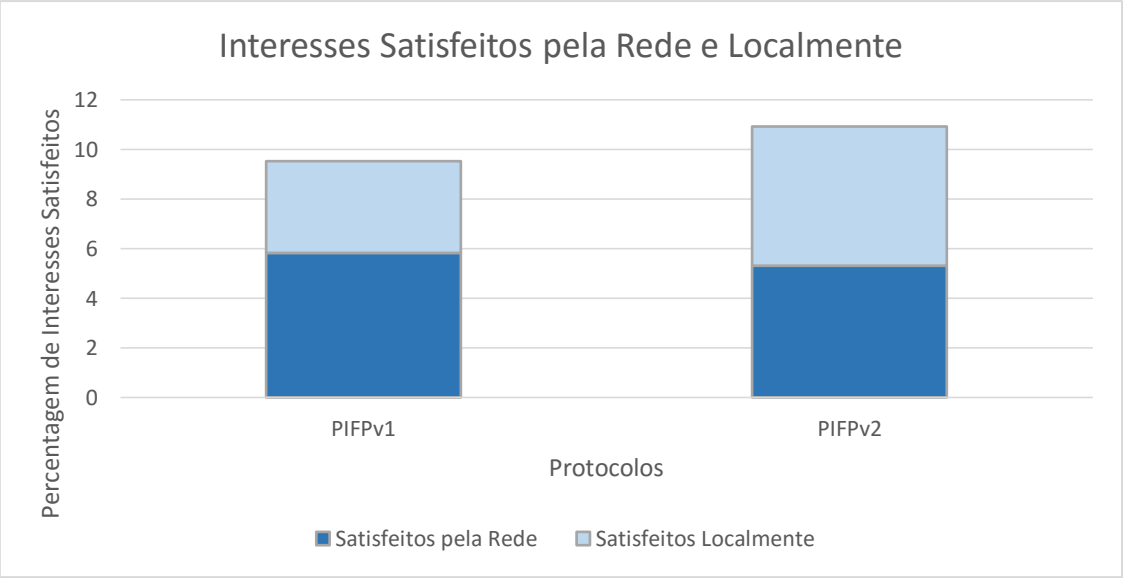


Figura 27 – Percentagem de Interesses Satisfeitos pela Rede e Localmente - Cenário 2

No que diz respeito ao cenário 2, através da Figura 27 pode ver-se distribuição dos interesses satisfeitos pela rede e localmente. No PIFPv1 61% dos interesses foram satisfeitos pela rede e 39% foram satisfeitos localmente. Já no PIFPv2 48% dos interesses foram satisfeitos através de pacotes chegados da rede e 52% foram satisfeitos através de conteúdo armazenado no próprio nó.

Tal acontece porque as mensagens têm muita dificuldade em propagar-se através da rede até atingir o seu destino final. Uma vez que os contactos são mais escassos (o que consequentemente dificulta a transmissão de mensagens), os nós ficam mais dependentes dos dados que têm armazenados.

O facto de se armazenar conteúdos pode ser sinónimo de satisfação de interesses. Ao armazenar dados, um nó poderá posteriormente responder com mensagens de dados a interesses que lhe sejam enviados, como também satisfazer os seus próprios interesses caso surjam. Isto contribui para aumentar o desempenho do protocolo.

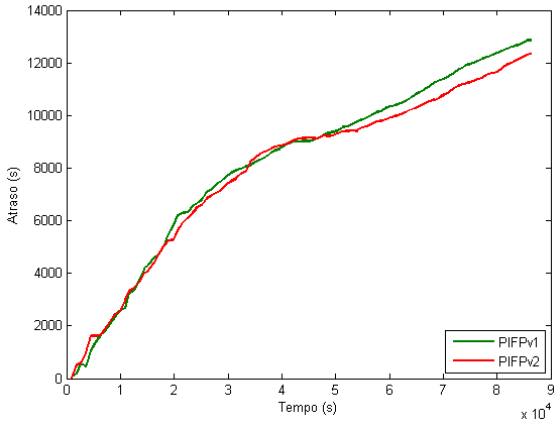


Figura 28 – Atraso Médio - Cenário 1

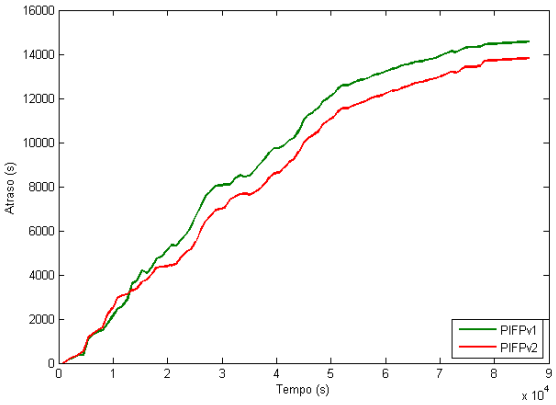


Figura 29 – Atraso Médio - Cenário 2

Um outro parâmetro para medir o desempenho de um protocolo é o atraso na entrega das mensagens. Através das Figura 28 e Figura 29 é possível verificar qual o atraso ao longo do tempo.

Nestes gráficos apenas são contemplados os atrasos para os interesses satisfeitos por pacotes vindos da rede. Caso um nó possua conteúdos na sua *cache* capaz de satisfazerem um interesse, este é imediatamente satisfeito e, portanto, não conta para esta estatística.

No cenário 1, o atraso é menor quando comparado com o cenário 2. Tal deve-se à ausência de contactos. A densidade de nós influencia o número de contactos. Quanto mais densa for a rede, mais contactos existirão. Um aumento de contactos significa um aumento de transmissões. Existindo menos transmissões os nós têm de aguardar até se voltarem a conectar com outro nó para poder encaminhar mensagens. Todo este processo resulta em atrasos maiores para redes menos densas. Para além disso, numa rede em que as mensagens podem ser encaminhadas por vários caminhos acelera o processo de entrega. Com mais caminhos, o trânsito de mensagens é mais fluído.

No cenário 1, o PIFPv1 e o PIFPv2 inicialmente têm o mesmo atraso, contudo, mais ou menos a meio da simulação, o atraso do PIFPv2 estabiliza por algum tempo. Até esse ponto, as probabilidades de encontrar interessados ainda não haviam convergido, ou seja, ainda não havia muito conhecimento acerca de interessados. O encaminhamento de mensagens de dados era escasso. Por isso é que ambas as versões apresentam praticamente o mesmo atraso. A partir do momento em que as probabilidades convergem, o atraso estabiliza durante um breve período. Depois volta a subir devido às *caches* começarem a ficar cheias, não podendo armazenar novos conteúdos para satisfazer novos interesses.

Contudo, em ambos os cenários, a versão 2 do PIFP é mais rápida a satisfazer os interesses que vão surgindo.

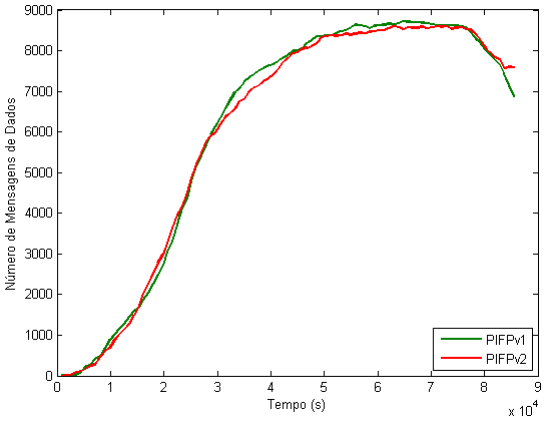


Figura 30 – Número de Mensagens de Dados - Cenário 1

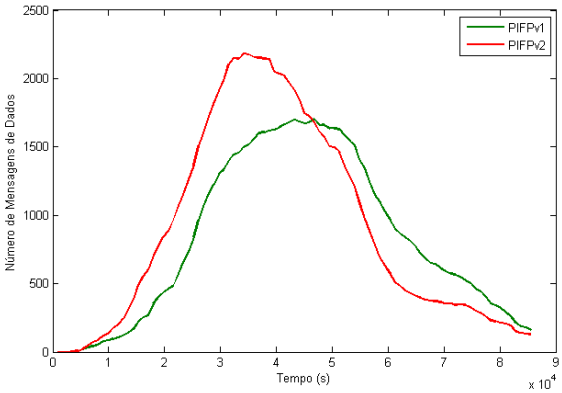


Figura 31 – Número de Mensagens de Dados - Cenário 2

Analisando os gráficos representados nas Figura 30 e Figura 31 é possível perceber a quantidade de dados que estão a circular na rede ao longo do tempo.

No cenário 1 o número de pacotes de dados na rede é bastante superior ao cenário 2. Tal deve-se ao facto de no cenário 1, devido à abundância de conexões, as mensagens podem ser encaminhadas com relativa facilidade. Desta feita, os conteúdos propagam-se pela rede. No cenário 2 os conteúdos não têm tantas oportunidades para serem encaminhados. Por conseguinte, existem menos mensagens de dados na rede.

Na versão 1 do protocolo PIFP, as mensagens de dados apenas são encaminhadas caso o outro nó manifeste interesse em recebê-los. Na versão 2, para além da abordagem implementada no PIFPv1, os conteúdos são também encaminhados para outro nó se esse outro nó tiver uma probabilidade maior de encontrar interessados. O PIFPv2 tem uma versão mais proactiva no que respeita à distribuição de conteúdos. Assim, à partida como existe maior replicação de dados, a quantidade dos mesmos a circular na rede deveria ser mais elevada que na versão 1. Contudo, caso um nó não contacte interessados num conteúdo, as probabilidades de encontrar interessados para o mesmo vão envelhecendo. Assim que atinge um valor mínimo, esse conteúdo é eliminado da CS. Desta forma, é reduzido o número de pacotes de dados a circular na rede. Este equilíbrio, entre conteúdos replicados e conteúdos eliminados, faz com que a quantidade de dados não dispare, como seria de esperar.

No cenário 2, tal como referido anteriormente, existe uma escassez de contactos. No PIFPv2, existe uma replicação de conteúdos, daí o aumento verificado no início da simulação. Contudo, como entre um contacto e outro passa muito tempo, assim que as probabilidades são atualizadas muitas delas atingem o limite mínimo e como consequência são eliminados os conteúdos correspondentes. Por outro lado, o tempo de vida dos pacotes de dados recebidos inicialmente expira e os dados são eliminados. Estes dois fatores em simultâneo resultam na queda acentuada verificada mais ou menos a meio do tempo de simulação.

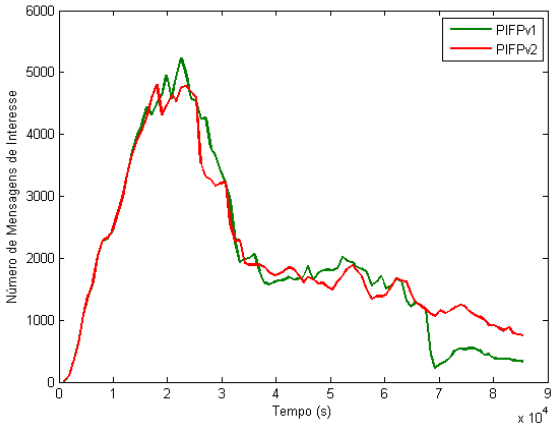


Figura 32 – Número de Mensagens de Interesse - Cenário 1

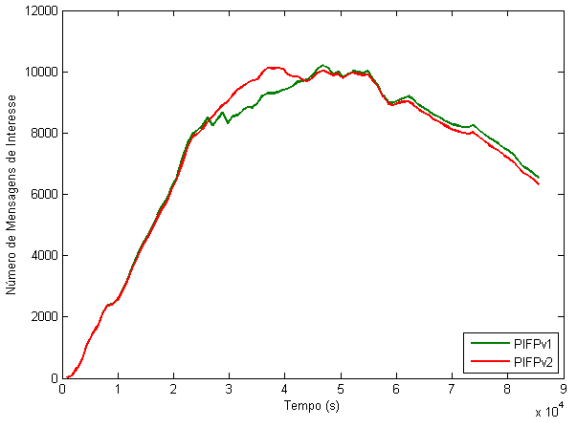


Figura 33 – Número de Mensagens de Interesse - Cenário 2

Por fim, resta analisar a quantidade de mensagens de interesse a circular na rede. Através das Figura 32 e Figura 33 é possível fazer essa análise.

No cenário 2 o número de mensagens de interesse é bastante superior ao cenário 1. Tal acontecimento deve-se ao facto de existirem menos interesses satisfeitos no cenário 2. Ao não serem satisfeitos, os interesses ficam na rede à espera de atingir um nó com o conteúdo pretendido. Contudo, o encaminhamento das mensagens é bastante lento devido aos motivos explicados anteriormente.

No cenário 1, existe um pico inicial de mensagens interesse. Porém, à medida que o tempo vai passando, esses interesses vão sendo satisfeitos. Consequentemente, o número de pacotes de interesse vai baixando também.

Em ambos os cenários, as duas versões apresentam comportamentos bastante semelhantes, pois utilizam a mesma estratégia de encaminhamento de mensagens de interesse.

7.3.2. Tamanho das Mensagens

Nesta fase da apresentação dos resultados pretende-se perceber até que ponto o tamanho das mensagens tem influência no desempenho dos protocolos. Serão apresentados resultados para tamanhos de mensagens entre os 50 e os 150k, entre os 250 e os 350k e, finalmente, entre os 750 e os 850k. Para efetuar estas simulações foi utilizado o cenário 1, descrito anteriormente.

7.3.2.1. Tamanho entre 50k – 150k

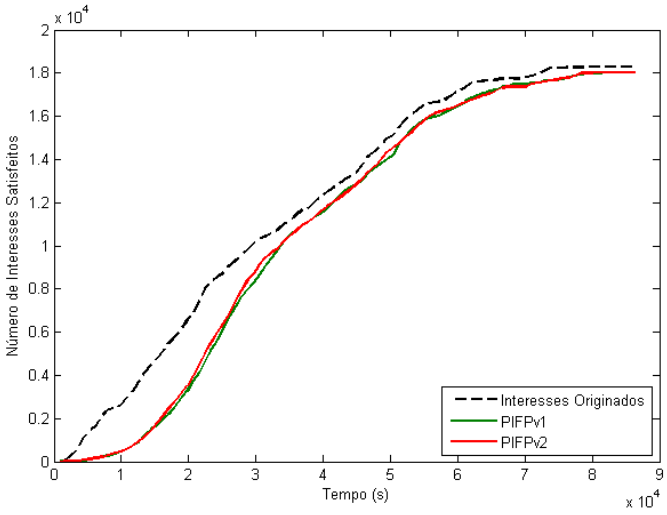


Figura 34 - Interesses Satisfeitos (50k -150k)

Através da Figura 34 é possível verificar que para mensagens de tamanho reduzido ambas as versões do protocolo apresentam um bom desempenho. Pode ver-se que quase todos os interesses que surgem são satisfeitos. A versão 1 do protocolo PIFP apresenta uma taxa de satisfação de 98%. O PIFPv2 apresenta, também, uma taxa de satisfação de 98%.

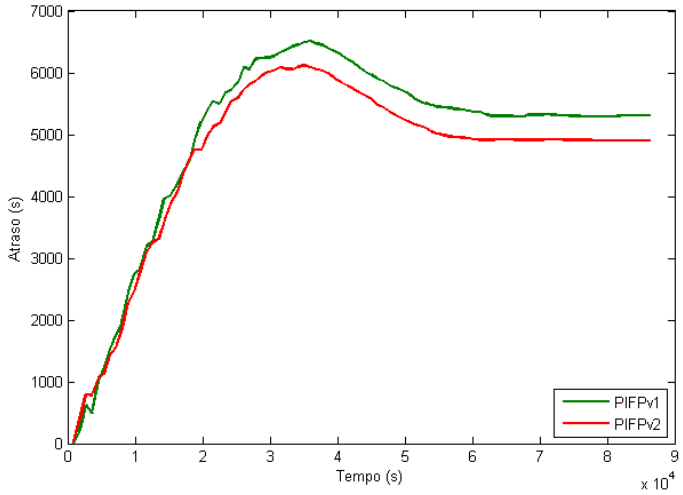


Figura 35 - Atraso Médio (50k-150k)

No que diz respeito ao atraso na entrega dos pacotes, pode ver-se, através da Figura 35, que para mensagens de tamanho pequeno o atraso é menor. É possível verificar também que no geral a versão 2 do PIFP é ligeiramente mais rápida na entrega dos conteúdos. Tal acontece devido ao facto dos conteúdos se encontrarem armazenados em nós com elevada probabilidade de encontrar interessados nos mesmos. Desta feita, aquisição de conteúdos para satisfazer um interesse é mais rápida.

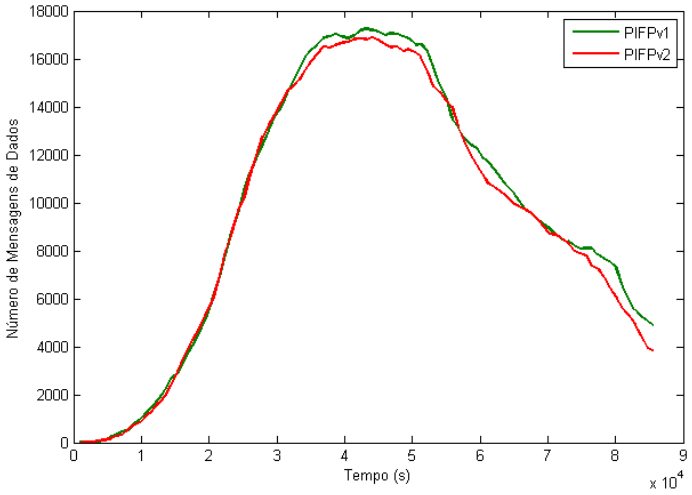


Figura 36 - Número de Mensagens de Dados (50k-150k)

Recorrendo à Figura 36 é possível verificar que o número de mensagens de dados a circular na rede é mais ou menos o mesmo nas duas versões do protocolo. Pode ver-se que mais ou menos a meio do tempo de simulação, por volta dos 44 mil segundos, o número de

mensagens de dados a circular começa a reduzir. Tal deve-se ao facto de vários nós possuírem vários conteúdos e podem satisfazer interesses diretamente, sem que seja necessário replicar conteúdos. Para além disso o tempo de vida das mensagens expira e estas são eliminadas da rede.

No PIFPv2 a carga de dados é ligeiramente menor. Isto porque, tal como havia sido referido, quando um nó deixa de contactar com interessados em determinado conteúdo, a probabilidade envelhece. Quando atinge o limite de 0,15 o conteúdo é eliminado uma vez que a probabilidade de encontrar interessados é reduzida.

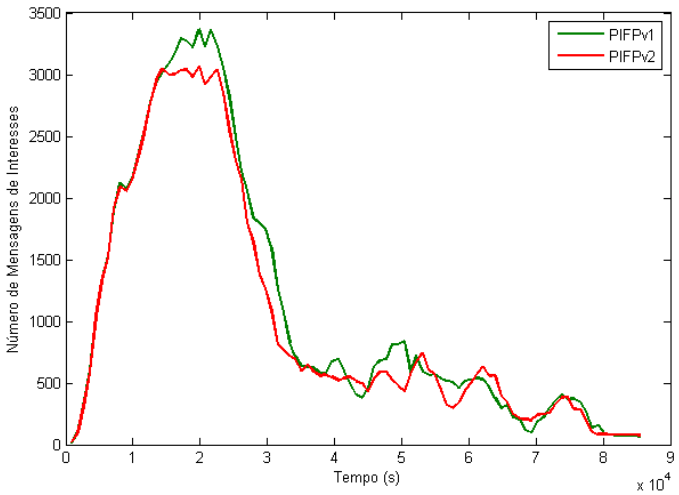


Figura 37 - Número de Mensagens de Interesse (50k-150k)

Na Figura 37 pode ver-se o número de mensagens de interesse a circular na rede ao longo do tempo de simulação. A partir dos 2000 segundos a carga de interesses na rede diminui de forma abrupta. Tal deve-se ao facto de cada um dos nós possuir conteúdos na sua CS que podem satisfazer localmente os seus interesses. Não é necessário assim enviar mensagens de interesse para a rede em busca do conteúdo pretendido.

Ambos os protocolos seguem praticamente a mesma linha, com o PIFPv2 a apresentar, geralmente, um menor número de interesses a circular na rede.

7.3.2.2 Tamanho entre 250k – 350k

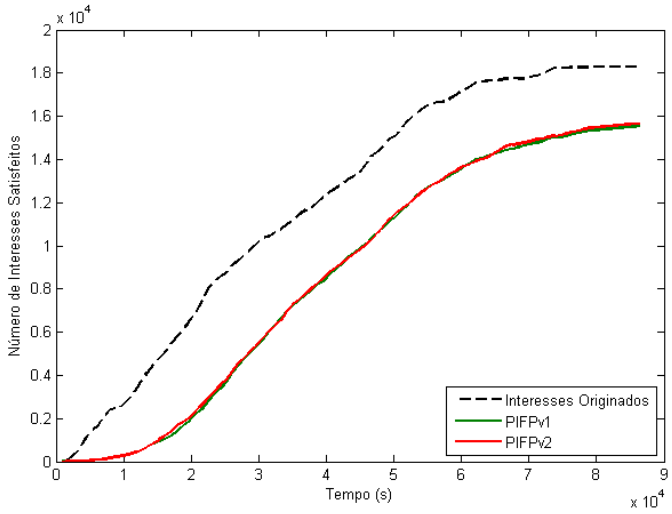


Figura 38 - Interesses Satisfeitos (250k-350k)

Tal como acontece para as mensagens de menor tamanho, tanto o PIFPv1 como o PIFPv2 apresentam resultados semelhantes. A versão 1 foi capaz de satisfazer 85% dos interesses originados. Já a versão 2 do protocolo satisfez 86% dos interesses.

Comparando com os resultados anteriores, pode ver-se que a taxa de satisfação baixou consideravelmente. Uma vez que o tamanho das mensagens aumentou, os nós não têm a capacidade de armazenar tantas mensagens. Isto leva a que seja mais difícil encontrar o conteúdo correto para satisfazer um determinado interesse. Existem menos mensagens espalhadas pela rede.

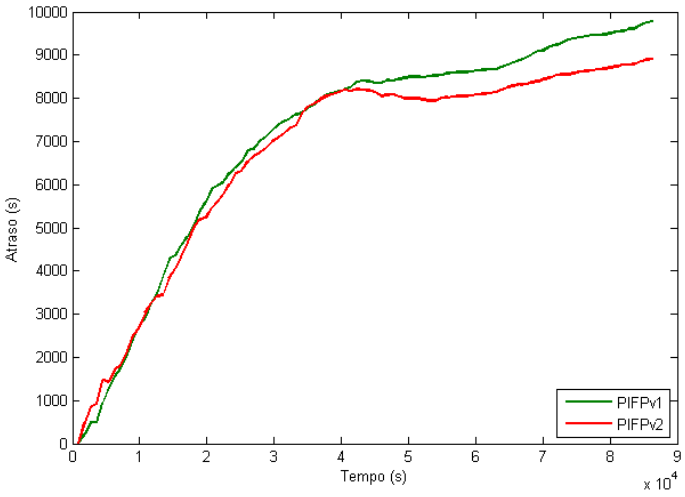


Figura 39 - Atraso Médio (250k-350k)

Através da Figura 39 é possível verificar o atraso médio na entrega das mensagens ao longo do tempo. Relativamente às mensagens mais pequenas, pode ver-se que existe uma maior demora na entrega das mensagens.

É possível ainda verificar que o PIFPv2 é mais rápido na entrega das mensagens, apresentando um atraso menor comparativamente ao PIFPv1. Mais uma vez, o facto de espalhar conteúdos pela rede acelera o processo de satisfação de interesses.

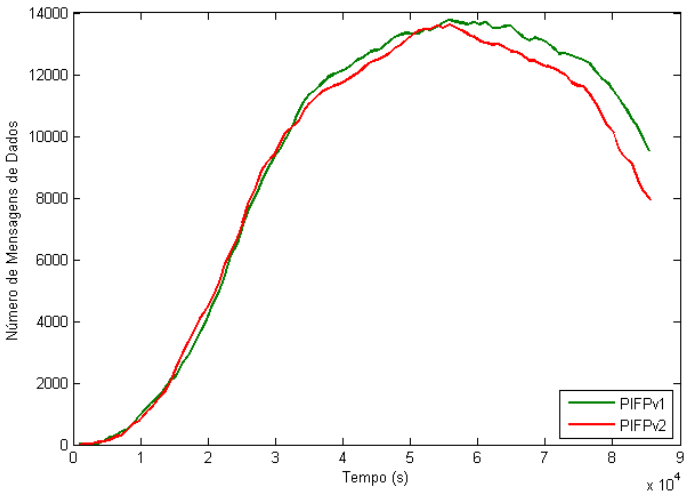


Figura 40 - Número de Mensagens de Dados (250k-350k)

Relativamente à quantidade de mensagens de dados a circular na rede, os resultados de simulação estão representados na Figura 40. A carga de dados na rede baixou quando comparado com o resultado utilizando mensagens de menor tamanho. O facto de a capacidade de armazenamento dos nós ser a mesma e o tamanho das mensagens ter aumentado faz com que não seja possível armazenar tantos conteúdos.

É possível verificar que o PIFPv2 necessita de menos dados na rede para satisfazer um maior número de interesses.

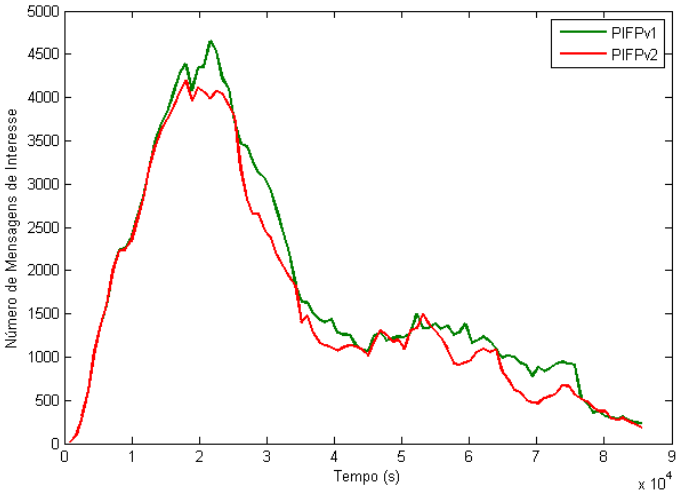


Figura 41 - Número de Mensagens de Interesse (250k-350k)

Recorrendo à Figura 41 pode ver-se o número de interesses ao longo do tempo. Em comparação com o teste anterior, com as mensagens mais pequenas, verifica-se um aumento do número de interesses. Tal deve-se ao facto de terem sido satisfeitos menos interesses.

O PIFPv2, tal como havia acontecido anteriormente, tem uma carga de mensagens de interesse menor. Isto porque existem mais interesses satisfeitos localmente.

7.3.2.3. Tamanho entre 750k – 850k

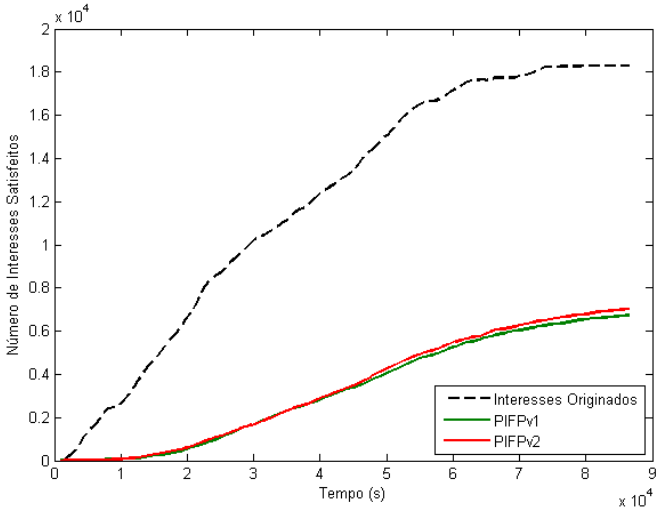


Figura 42 - Interesses Satisfeitos (750k-850k)

Analisando o gráfico da Figura 42, é possível verificar que a taxa de satisfação de interesses desceu bastante quando comparada com os testes anteriores. Mais uma vez, o facto do tamanho das mensagens ter aumentado não permite que se guardem várias mensagens nos nós, o que dificultada obtenção de conteúdos.

A versão 1 do PIFP apresenta uma taxa de satisfação de 36%, e a versão 2 apresenta uma taxa de 38%.

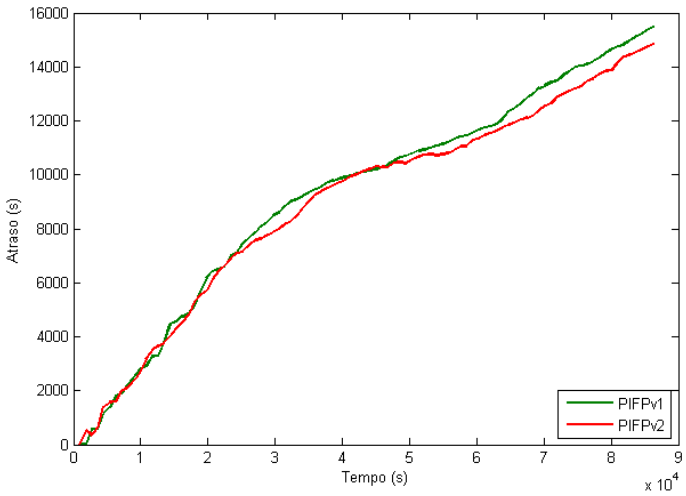


Figura 43 - Atraso Médio (750k-850k)

Tal como era esperado, o atraso aumentou comparativamente aos testes anteriores. Isto acontece devido aos motivos anteriormente expostos. Com o aumento das mensagens, os *buffers* dos nós enchem mais rapidamente e armazenam menos mensagens. Assim, existem menos conteúdos na rede o que dificulta a sua obtenção.

Apesar de tudo, a versão 2 do PIFP apresenta um atraso menor, quando comparado com o PIFPv1.

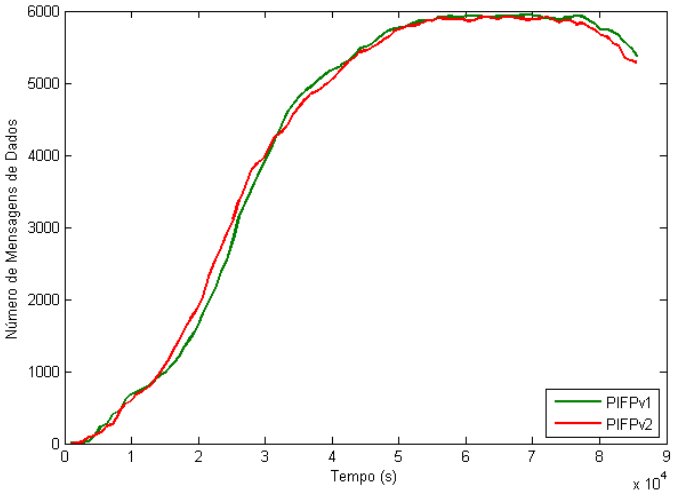


Figura 44 - Número de Mensagens de Dados (750k-850k)

No que respeita à carga de dados na rede, através da Figura 44 é possível verificar que existem menos dados a circular na rede. Tal como referido anteriormente, o aumento do tamanho das mensagens limita o número de mensagens que podem ser armazenadas.

Ambos protocolos apresentam praticamente a mesma linha. Na versão 2 do protocolo, os conteúdos não podem ser eliminados da CS pois existindo um menor número de conteúdos armazenados, é mais difícil que cada probabilidade baixe até ao limite.

Por exemplo, se existirem 30 conteúdos armazenados na CS, é mais fácil que vários deles baixem a probabilidade de encontrar interessados até atingirem o limite mínimo e serem eliminados. Se existirem apenas 10 conteúdos armazenados, apenas um ou outro baixam a sua probabilidade até ao limite e são eliminados.

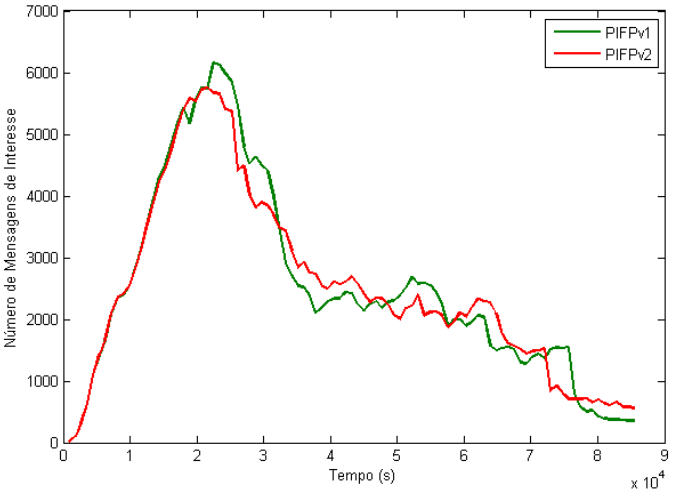


Figura 45 - Número de Mensagens de Interesse (750k-850k)

Relativamente às mensagens de interesse, através da análise do gráfico da Figura 45 pode ver-se que o número de interesses aumentou. Tal como aconteceu nos testes anteriores, com o aumento das mensagens de 50k-150k para os 250k-350k, o número de interesses insatisfeitos aumentou. Este facto implica que existam mais pacotes de interesse na rede em busca dos conteúdos.

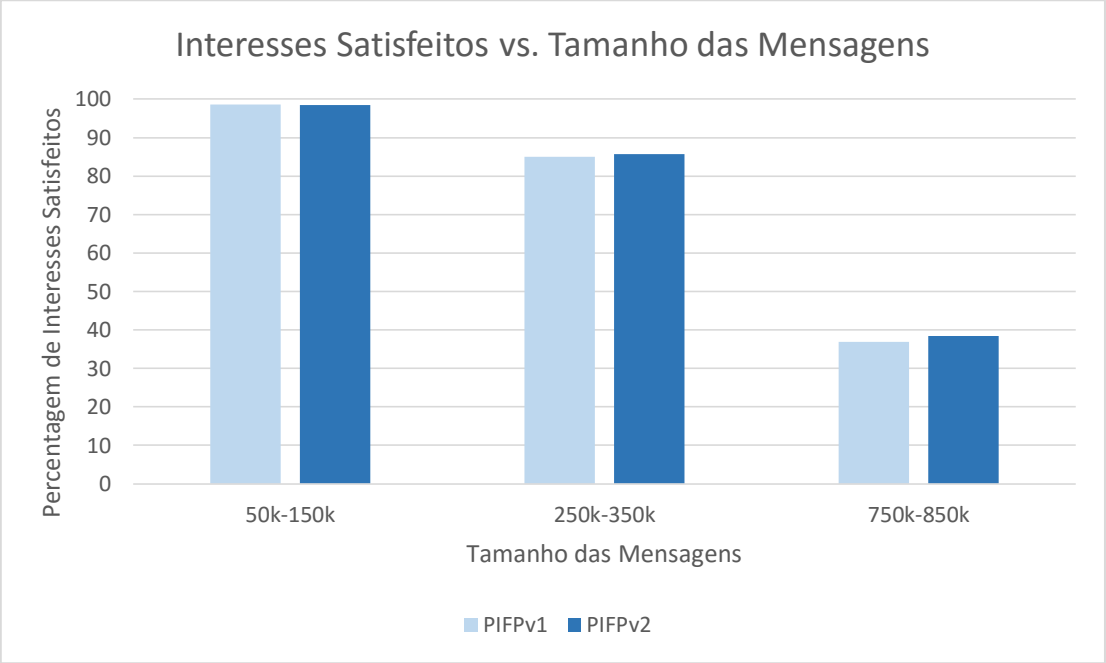


Figura 46 - Impacto do Tamanho das Mensagens na Satisfação de Interesses

Tal como se pode verificar através do gráfico presente na Figura 46, o tamanho das mensagens influencia bastante a satisfação dos interesses. Pode ver-se que com o aumento do tamanho das mensagens a percentagem de interesses satisfeitos baixa bastante, independentemente da versão do PIFP utilizada. Com o aumento do tamanho das mensagens, e a mesma capacidade de armazenamento, não é possível armazenar tantas mensagens. Deste modo, perde-se assim capacidade de encaminhar os conteúdos recebidos uma vez que não há espaço para os armazenar. Por outro lado, neste tipo de redes oportunistas, as conexões entre os nós são muito instáveis. Durante uma conexão é possível encaminhar várias mensagens de tamanho mais pequeno. Sendo as mensagens maiores, o número de mensagens encaminhadas durante uma conexão é menor. Pode ainda acontecer de a conexão se perder a meio da transmissão e a mensagem não ser enviada na sua totalidade, sendo assim descartada.

Pode ver-se que para mensagens mais pequenas, ambas as versões do PIFP apresentam um desempenho semelhante. Para mensagens maiores o PIFPv2 tem um desempenho ligeiramente melhor.

7.3.3. Consumo de Energia

Um outro fator a ter em conta nas DTNs é a energia dos nós. Em muitos casos, neste tipos de redes, os nós são alimentados por baterias e a poupança de energia deve ser eficaz. Deste modo não há necessidade de substituir o equipamento. Uma vez que o PIFP é um protocolo que utiliza dados nomeados nas DTNs deve haver preocupação quanto à energia gasta pelos nós.

Serão agora apresentados resultados que permitem analisar o consumo de energia. Foi utilizado o cenário 1 para efetuar estas simulações.

Tal como acontece na configuração dos cenários, também os gastos de energia podem ser definidos previamente à simulação. Para este teste foram utilizados os mesmos valores usados em [8].

Tabela 4 - Configurações de Consumo de Energia

Parâmetro	Valor
Energia Inicial	Produtores – 15000 Unidades Consumidores – 10000 Unidades
Consumo de Energia por Pesquisa	0.1 Unidades
Consumo de Energia por Transmissão	0.2 Unidades
Consumo de Energia por Resposta de Pesquisa	0.1 Unidades
Consumo de Energia Base	0.01 Unidades

Recorrendo à Tabela 4 podem ver-se as configurações iniciais no que diz respeito ao consumo de energia. Cada ação tem um gasto de energia associado. Inicialmente, os nós produtores têm 15000 unidades de energia e os nós consumidores têm 10000 unidades. Ao longo da simulação, consoante as suas ações, a energia que cada nó possui vai diminuindo.

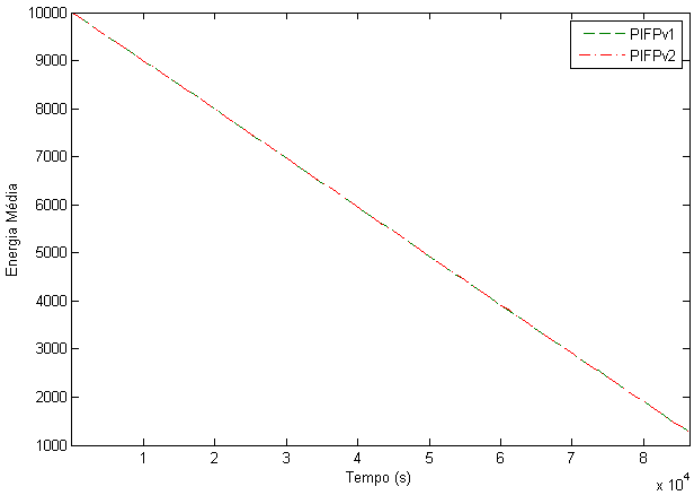


Figura 47 - Energia Média

Através da Figura 47 é possível verificar que em ambas as versões o consumo de energia é praticamente o mesmo.

Tabela 5 - Nível de Energia Médio ao Longo do Tempo

Tempo (s)	PIFPv1	PIFPv2
Inicio	9999	9999
28800 (8 horas)	7093	7094
57600 (16 horas)	4161	4162
86400 (24 horas)	1264	1264

Na versão 2 do PIFP existe um maior número de transmissões de conteúdos, o que deveria significar um maior consumo de energia. Contudo, existe também um maior número de interesses satisfeitos localmente, sem ser necessária a transmissão de conteúdos. É por este facto que ambas as versões apresentam um consumo de energia tão idêntico como se pode confirmar através da Tabela 4.

7.3.4. Número de Saltos

Por fim, resta analisar o número de saltos que cada mensagem faz até chegar ao seu destino. O número de saltos corresponde ao número de nós pelo qual uma mensagem passou, menos um, visto que o primeiro nó não conta.

Primeiramente é analisado o número de saltos das mensagens de dados. Apenas os dados entregues no destino final são considerados.

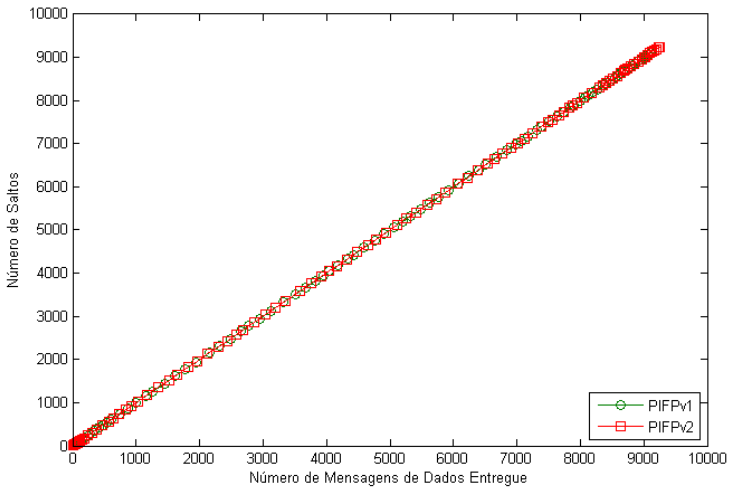


Figura 48 - Número de Saltos (Mensagens de Dados)

Através da análise do gráfico da Figura 48 pode ver-se que ambas as versões apresentam praticamente o mesmo número de saltos. Inicialmente era esperado que o PIFPv2, devido à sua natureza semi-proactiva, tivesse um maior número de saltos. Contudo tal não acontece pois as probabilidades de encontrar interessados demoram algum tempo a convergir. Nos instantes iniciais não há muito conhecimento de interessados. Ao longo do tempo, esse conhecimento vai aumentando e as mensagens de dados vão sendo enviadas para outros nós com maior probabilidade de encontrar interessados. Contudo, o número de saltos não aumenta pois os conteúdos ficam mais "próximos" do seu destino final. Existe um equilíbrio entre o aumento do número de saltos provocado pela replicação de conteúdos e o facto de as mensagens de dados não necessitarem de passar por tantos nós para chegar ao destino final.

Tabela 6 - Número de Saltos de Mensagens de Dados

Protocolo	Média	Mediana	Máximo	Mínimo
PIFPv1	1,015	1	3	1
PIFPv2	1,02	1	3	1

Através da Tabela 6 pode ver-se que ambas as versões do protocolo se comportam de forma bastante semelhante no que diz respeito ao número de saltos. Apesar da sua natureza proactiva, na distribuição de conteúdos, o PIFPv2 consegue manter o número de saltos comparativamente ao PIFPv1.

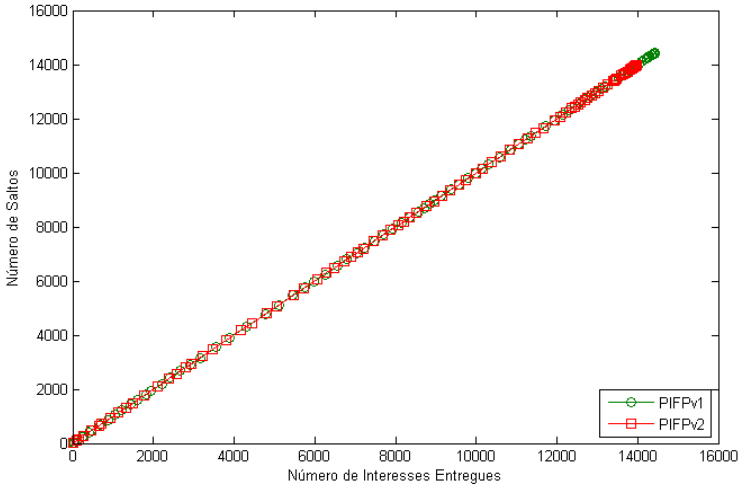


Figura 49 - Número de Saltos (Mensagens de Interesse)

Relativamente ao número de saltos das mensagens de interesse pode ver-se, através do gráfico da Figura 49, a distribuição do número total de saltos em relação aos interesses entregues.

Em ambas as versões do protocolo PIFP a estratégia de encaminhamento de mensagens de interesse é a mesma. Por este motivo, o comportamento de ambas as versões é muito semelhante.

No PIFPv2 existe um menor número total de saltos, isto porque são enviados menos pacotes de interesse. Existem mais interesses satisfeitos localmente, sem que haja necessidade de enviar mensagens a pedir conteúdos.

7.4. Análise e Discussão Crítica dos Resultados

Através dos testes realizados anteriormente pôde avaliar-se até que ponto a versão 2 do protocolo PIFP superava a primeira versão. Após uma verificação e análise dos resultados obtidos é possível afirmar que existem de facto melhorias

No primeiro cenário, o cenário com maior densidade de nós, o PIFPv2 consegue satisfazer mais interesses quando comparado com o PIFPv1. Também no cenário 2, onde a densidade de nós é bastante mais baixa, o PIFPv2 supera a primeira versão no que diz respeito à taxa de satisfação de interesses. Ambas as versões apresentam um desempenho bastante inferior no cenário 2 quando comparado com o cenário 1. O facto de a rede apresentar menor densidade

tem repercussões na taxa de satisfação de interesses e nenhuma das versões tem a capacidade de contrariar essa limitação.

No que respeita ao atraso na entrega dos conteúdos, os resultados das simulações indicam que o PIFPv2 tem atrasos menores. Em ambos os cenários, a versão 2 foi mais rápida a satisfazer os interesses que iam surgindo. Contudo, no cenário 2, o atraso sobe em ambas as versões quando comparado com os resultados do cenário 1.

Relativamente à carga de interesses as duas versões apresentam comportamentos semelhantes. Isto porque ambas as versões utilizam a mesma estratégia para o encaminhamento de mensagens de interesse.

Quanto ao número de pacotes de dados a circular na rede os resultados revelam que o PIFPv2 recorre a um número igual ou inferior de dados para satisfazer um maior número de interesses quando comparado com o PIFPv1. Em raros momentos a carga de dados do PIFPv2 é superior à do PIFPv1. No cenário 2, inicialmente o PIFPv2 tem uma grande carga de dados devido à sua natureza proactiva. Contudo, quando as probabilidades de encontrar interessados são atualizadas esse número desce bastante. São eliminados vários conteúdos da CS dos nós, uma vez que a probabilidade de encontrar interessados nos mesmos é reduzida.

Na primeira versão do PIFP, apenas existia transmissão de conteúdos caso um nó recebesse uma mensagem de interesse proveniente de outro nó. Não existiam pacotes transferidos a não ser que fosse mesmo necessário. Apenas era enviado um pacote de interesse porque um nó tinha necessidade de um conteúdo, e apenas era transmitido um pacote de dados porque havia a necessidade de responder a um interesse recebido. Apesar disso, conseguiu-se reduzir o número de pacotes da rede recorrendo à eliminação de conteúdos que não eram uteis para a rede. Passou a eliminar-se pacotes cuja probabilidade de encontrar interessados era reduzida.

Nos testes realizados com mensagens de tamanhos diferentes, foi possível verificar que este é um fator que influência bastante o desempenho de um protocolo. Quanto maiores forem as mensagens, pior será o desempenho do protocolo a ser testado.

Com mensagens de tamanho curto, ambas as versões apresentaram uma taxa de satisfação perto dos 100%. Contudo, o PIFPv2 apresenta atrasos mais curtos na satisfação dos mesmos, bem como uma carga de dados menor, quando comparado com o PIFPv1.

Com o aumento do tamanho das mensagens a taxa de satisfação vai reduzindo. Porém, ainda assim o PIFPv2 tem uma taxa de satisfação superior ao PIFPv1. O atraso médio é também menor no PIFPv2. A carga de dados para satisfazer interesses que vão surgindo é menor no PIFPv2 do que no PIFPv1.

Relativamente ao número de saltos, ambos os protocolos se comportam de forma idêntica. Tanto nos pacotes de dados como nos pacotes de interesse, os PIFPv1 e o PIFPv2 apresentam um comportamento bastante semelhante.

Em termos de energia gasta, ambas as versões do PIFP apresentam os mesmos gastos de energia, não havendo praticamente distinção.

Os resultados obtidos demonstram que apesar de existirem efetivamente melhorias estas não são muito acentuadas. Em muitos aspetos ambas as versões do protocolo PIFP apresentam o mesmo comportamento. Alguns dos resultados não foram também de acordo com o esperado.

No que respeita ao atraso médio, era de esperar que com o aumento de conteúdos espalhados ao longo da rede o atraso médio diminuísse e acabasse por estabilizar. Contudo tal não se verifica na totalidade, não pelo menos em todos os cenários testados (apenas na simulação para mensagens de tamanho reduzido os resultados foram de acordo com o esperado). O atraso diminui, efetivamente, contudo apresenta sempre tendência a aumentar. Isto acontece devido ao facto de com passar do tempo as *caches* dos nós ficarem mais cheias e perdem a capacidade de armazenar novos conteúdos para satisfazer novos interesses. É necessário aguardar que sejam eliminados conteúdos da CS para que se possam armazenar novos conteúdos e satisfazer novos pedidos que vão surgindo.

Quanto ao número de saltos, era de esperar que este parâmetro diminuísse. Com a versão 2 do PIFP existe uma replicação de conteúdos. Existem mais conteúdos espalhados pela rede e era de esperar que os conteúdos ficassem mais "próximos" dos interessados. Posto isto, deveria ser necessário efetuar menos saltos para que os interesses fossem satisfeitos. Porém, tal não se verifica.

Relativamente ao consumo de energia, era de esperar que na versão 2 do protocolo se gastasse mais energia, uma vez que existem mais transmissões. Contudo, o facto de existirem mais conteúdos armazenados em *cache* ao longo da rede possibilita que mais interesses sejam satisfeitos localmente, sem que haja necessidade de se proceder a uma transmissão de pacotes.

Tabela 7 - Tabela de Resumo dos Resultados Obtidos

			PIFPv1	PIFPv2
Densidade da Rede	Cenário 1	Interesses Satisfeitos	63%	65%
		Atraso	Maior	Menor
		Carga da Rede	Semelhante	Semelhante
	Cenário 2	Interesses Satisfeitos	9%	11%
		Atraso	Maior	Menor
		Carga da Rede	Inicialmente possui menos dados	Inicialmente possui mais dados
Tamanho das Mensagens	50k – 150k	Interesses Satisfeitos	98%	98%
		Atraso	Maior	Menor
		Carga da Rede	Maior número de dados	Menor número de dados
	250k – 350k	Interesses Satisfeitos	85%	86%
		Atraso	Maior	Menor
		Carga da Rede	Maior número de dados	Menor número de dados
	750k – 850k	Interesses Satisfeitos	36%	38%
		Atraso	Maior	Menor
		Carga da Rede	Semelhante	Semelhante
Energia Utilizada			Semelhante	Semelhante
Número de Saltos		Mensagens de Dados	Média 1,01	Média 1,02
		Mensagens de Interesse	Média 1	Média 1

Capítulo 8 – Conclusões e Trabalho Futuro

Ao longo da realização desta dissertação foram surgindo vários desafios.

Começou-se por uma componente teórica, de pesquisa e estudo sobre as DTNs e NDNs. O primeiro desafio foi perceber até que ponto duas arquiteturas à partida tão distintas poderiam operar no mesmo ambiente. Perceber como retirar o melhor de cada uma delas em prol de um melhor desempenho da rede.

Por um lado, as DTNs foram desenvolvidas para operar em cenários onde as ligações fim-a-fim nem sempre estão disponíveis. Num momento uma conexão pode estar disponível e no momento seguinte pode perder-se. Esta arquitetura possui uma série de mecanismos e paradigmas que permitem lidar com a incerteza do estado das conexões. Desde o paradigma de *store-carry-and-forward* até à fragmentação de mensagens, passando por um mecanismo de confiabilidade como a transferência de custódia.

Por outro lado, as NDNs têm características que lhe permitem lidar com a mobilidade dos nós. Esta foca-se nos dados que são transmitidos e não na sua localização. Desta forma, existe uma maior compreensão dos dados e permite a sua posterior reutilização para satisfazer pedidos futuros.

Percebeu-se então que a integração destas duas arquiteturas era um caminho viável e que poderia trazer benefícios a nível de desempenho da rede. Nas DTNs existe um certo egoísmo por parte dos nós. Ou seja, assim que um nó não tem capacidade para armazenar mais mensagens, começa a eliminar as que possui. Com a integração das NDNs, devido às suas caraterísticas, existe uma melhor compreensão da informação que é transmitida. Desta forma, podem selecionar-se os dados a serem eliminados criando um menor impacto na rede.

Existiu também uma componente prática que visava o melhoramento de um protocolo de encaminhamento que lida com dados nomeados mas que é capaz de operar em redes oportunistas. Trata-se do PIFP, sugerido em [8]. Este era o grande desafio proposto nesta dissertação.

O PIFP, inspirado no PRoPHET, um protocolo de encaminhamento para DTNs, é capaz de lidar com dados nomeados e explorar a frequência com que os vários contactos entre os nós

ocorrem. Deste modo, as mensagens são encaminhadas tendo em conta essa frequência de contacto.

O PIFP anteriormente desenvolvido apenas tinha em conta a frequência de contacto de um nó com um determinado conteúdo. Apenas os pacotes de interesse eram encaminhados tendo em conta probabilidades. Os conteúdos apenas eram encaminhados em resposta a um interesse recebido.

Foi proposta uma estratégia de encaminhamento de conteúdos baseada na probabilidade de encontrar interessados. Depois de uma explicação detalhada dessa proposta de melhoramento, procedeu-se à sua implementação na plataforma ICONE.

Com os melhoramentos feitos, o PIFP, é agora capaz de encaminhar também pacotes de conteúdo tendo em conta a probabilidade de encontrar interessados.

Recorrendo a simulações foram testadas essas melhorias e foi possível perceber até que ponto o desempenho do protocolo havia melhorado. A nível da taxa de satisfação de interesses houve uma ligeira melhoria. A nova versão do protocolo PIFP é capaz de satisfazer um maior número de interesses quando comparada com a versão anterior.

O grande melhoramento encontra-se ao nível dos atrasos e do número de pacotes a circular na rede. Após a realização das simulações e da recolha e análise dos resultados percebeu-se que os atrasos na satisfação de interesses era menor na nova versão do protocolo. Os conteúdos que solicitam determinado conteúdo vêm as suas necessidades satisfeitas mais rapidamente.

Por outro lado, a carga de mensagens de dados na rede é geralmente menor. Em alguns casos a carga é semelhante ou a diminuição da mesma é pouco significativa. Contudo, apenas por alguns instantes é superior, dado essa métrica volta a baixar alguns instantes depois.

Foram testadas as duas versões em dois cenários distintos. Um deles com uma alta densidade de nós e outro com uma densidade mais baixa. Em ambos os cenários, o desempenho do PIFPv2 foi superior. Tanto a nível de interesses satisfeitos, como a nível de atrasos e ainda a nível de congestionamento da rede.

Foram também feitos testes para tamanhos de mensagens diferentes. Em todos eles, o PIFPv2 apresentou melhor desempenho. Essas melhorias foram mais visíveis em situações com mensagens de tamanho maior.

Concluindo, o PIFPv2 trouxe melhorias visíveis ao desempenho da rede. Isto na medida em que satisfaz mais interesses, de forma mais rápida, com uma menor carga da rede.

Porém, existe ainda muito espaço para melhorias que podem aumentar ainda mais o desempenho deste protocolo.

Uma vez que o PIFP opera em cenários onde as conexões são bastante instáveis, seria importante definir uma estratégia de fragmentação de conteúdos. Como se pôde verificar, para mensagens de um tamanho maior, o desempenho do protocolo está longe de ser brilhante. Com uma estratégia de fragmentação das mensagens o nível de desempenho poderia aumentar bastante. Seria possível enviar uma mensagem, uma parte de cada vez, utilizando caminhos distintos. Desta forma, a taxa de satisfação de interesses poderia aumentar. O objetivo seria prever os tempos de contacto e efetuar uma fragmentação das mensagens de acordo com essa previsão.

Um outro melhoramento importante poderia ser feito a nível de gestão de *caches*. Seria importante estabelecer uma estratégia de gestão do espaço de armazenamento que selecionasse melhor quais os conteúdos que devem ser eliminados. Isto permitiria que os nós tivessem a capacidade de armazenar conteúdos importantes para si e para satisfazer futuros interesses.

Por fim, a elaboração de uma estratégia de escalonamento do tempo seria também uma melhoria importante. Numa rede oportunista os contatos podem ter uma duração mais curta que o desejado. Seria importante prever o tempo de contacto e definir que ações se devem ser priorizadas tendo em vista um melhor desempenho da rede.

Bibliografia

[1] K. Fall, “A delay-tolerant network architecture for challenged internets,” *Proc. 2003 Conf. Appl. Technol. Archit. Protoc. Comput. Commun. - SIGCOMM ’03*, p. 27, 2003.

[2] F. Warthman, “Delay- and Disruption-Tolerant Networks (DTNs) - A Tutorial,” p. 35, 2012.

[3] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, “Delay-Tolerant Networking Architecture,” *Rfc4838*, vol. 54, p. 258, 2007.

[4] G. Tyson, J. Bigham, and E. Bodanese, “Towards an Information-Centric Delay-Tolerant Network.”

[5] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, “Named Data Networking,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, 2014.

[6] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, L. Wang, P. Crowley, and E. Yeh, “Named Data Networking (NDN) Project,” *PARC Tech Rep.*, pp. 1–26, 2010.

[7] A. Keränen, “The ONE Simulator for DTN Protocol Evaluation,” *Proc. Second Int. ICST Conf. Simul. Tools Tech.*, p. 55, 2009.

[8] P. Alexandre and G. Duarte, *Paulo Alexandre Gomes Duarte Dados Nomeados para Redes Tolerantes a Atrasos*. 2014.

[9] E. De Engenharia, “Fábio Manuel Afonso Cerqueira Caching em Redes Tolerantes a Atrasos com Dados Nomeados Fábio Manuel Afonso Cerqueira Caching em Redes Tolerantes a Atrasos com Dados Nomeados,” 2014.

[10] A. Lindgren, A. Doria, E. Davies, and S. Grasic, “Probabilistic Routing Protocol for Intermittently Connected Networks,” *Rfc6693*, pp. 1–113, 2012.

[11] I. Chlamtac, M. Conti, and J. Liu, “Mobile ad hoc networking: imperatives and challenges,” *Ad Hoc Networks*, pp. 153–158, 2003.

[12] A. Rumpold, “Transmission Protocols for Delay-Tolerant Networks,” *Semin. Innov. Internettechnologien und Mobilkommunikation SS2011*, pp. 137–142, 2011.

[13] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss, “Delay-tolerant networking: An approach to interplanetary internet,” *IEEE Commun. Mag.*, vol. 41, no. 6, pp. 128–136, 2003.

[14] C. Caini, H. Cruickshank, S. Farrell, and M. Marchese, “Delay-and disruption-tolerant networking (DTN): An alternative solution for future satellite networking applications,” *Proc. IEEE*, vol. 99, no. 11, pp. 1980–1997, 2011.

[15] K. Scott and S. Burleigh, “Bundle Protocol Specification,” *Rfc5050*, pp. 1–50, 2007.

[16] P. Ginzboorg, V. Niemi, and J. Ott, “Message Fragmentation for Disrupted Links,” *World Wireless, Mob. Multimed. Networks (WoWMoM), 2012 IEEE Int. Symp.*, pp. 415–424, 2012.

[17] R. Fielding and L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax Status,” *Rfc3986*, pp. 1–61, 2005.

[18] M. J. Khabbaz, C. M. Assi, and W. F. Fawaz, “Disruption-tolerant networking: A comprehensive survey on recent developments and persisting challenges,” *IEEE Commun. Surv. Tutorials*, vol. 14, no. 2, pp. 607–640, 2012.

[19] X. Zhang, G. Neglia, J. Kurose, and D. Towsley, “Performance modeling of epidemic routing,” *Comput. Networks*, vol. 51, no. 10, pp. 2867–2891, 2007.

[20] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, “Spray and Wait : An Efficient Routing Scheme for,” *Direct*, pp. 252–259, 2005.

[21] T. Koponen, K. H. Kim, and S. Shenker, “A Data-Oriented (and Beyond) Network Architecture,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 181–192.

[22] D. Han, A. Anand, F. Dogar, B. Li, H. Lim, M. Machado, A. Mukundan, W. Wu, A. Akella, D. G. Andersen, J. W. Byers, S. Srinivasan, and P. Steenkiste, “XIA : Efficient Support for Evolvable Internetworking,” *Nsdi*, pp. 1–14, 2012.

[23] N. Fotiou, P. Nikander, D. Trossen, and G. C. Polyzos, “Developing information networking further: From PSIRP to PURSUIT,” *Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng.*, vol. 66 LNICST, pp. 1–13, 2012.

[24] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, “A case for stateful forwarding plane,” *Comput. Commun.*, vol. 36, no. 7, pp. 779–791, 2013.

[25] V. Jacobson, D. K. Smetters, J. D. Thornton, M. Plass, N. Briggs, and R. Braynard, “Networking named content,” *Commun. ACM*, vol. 55, no. 1, p. 117, 2012.

[26] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, K. V Katsaros, and G. C. Polyzos, “A Survey of Information-Centric Networking Research,” pp. 1–26, 2013.

[27] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, “Adaptive forwarding in named data networking,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 3, p. 62, 2012.

[28] A. Alyyan and L. Wang, “NLSR : Named-data Link State Routing Protocol,” pp. 15–20, 2013.

[29] J. Moy, “OSPF Version 2,” *Rfc2178*, pp. 1–211, 1997.

[30] L. Wang, a K. M. M. Hoque, C. Yiy, A. Alyyan, and B. Zhangy, “OSPFN : An OSPF Based Routing Protocol for Named Data Networking,” pp. 1–15, 2012.

[31] M. Tortelli, L. A. Grieco, G. Boggia, K. Pentikousis, and P. Bari, “COBRA : Lean Intra-domain Routing in NDN,” *Consum. Commun. Netw. Conf.*, pp. 853–858, 2014.

[32] A. Broder and M. Mitzenmacher, “Network Applications of Bloom Filters: A Survey,” *Internet Math.*, vol. 1, no. 4, pp. 485–509, 2004.

[33] Y. Lu, W. Cheng, L. Tung, and M. Gerla, “Social-tie based Content Retrieval for Delay-Tolerant Mobile Ad-hoc Networks,” 2013.

[34] Y. Lu, X. Li, Y. T. Yu, and M. Gerla, “Information-centric delay-tolerant mobile ad-hoc networks,” *Proc. - IEEE INFOCOM*, pp. 428–433, 2014.

[35] A. Ahmad and L. Dey, “A k-mean clustering algorithm for mixed numeric and categorical data,” *Data Knowl. Eng.*, vol. 63, no. 2, pp. 503–527, 2007.

[36] M. Meisel, “BOND : Unifying Mobile Networks with Named Data Freeform Wireless Networks,” *Wirel. Networks*, 2011.

[37] M. Meisel, V. Pappas, and L. Zhang, “Listen first, broadcast later: Topology-agnostic forwarding under high dynamics,” *Annu. Conf. Int. …*, pp. 1–8, 2010.

[38] Y. T. Yu, R. B. Dilmaghani, S. Calo, M. Y. Sanadidi, and M. Gerla, “Interest propagation in named data manets,” *2013 Int. Conf. Comput. Netw. Commun. ICNC 2013*, pp. 1118–1122, 2013.

[39] F. N. Dos Santos and B. Ertl, “CEDO: content-centric dissemination algorithm for delay-tolerant networks,” *Proc. 16th …*, pp. 377–386, 2013.

[40] H. Song, S. Dharmapurikar, J. Turner, and J. Lockwood, “Fast hash table lookup using extended bloom filter,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, p. 181, 2005.